Example-Based Water Animation

Pickup, David

Award date:
2013

Awarding institution:
University of Bath

Example-Based Water Animation

David Lemor Pickup

A thesis submitted for the degree of Doctor of Philosophy

University of Bath

Department of Computer Science

March 2013

Abstract

We present the argument that video footage of real scenes can be used as input examples from which novel three-dimensional scenes can be created. We argue that the parameters used by traditional animation techniques based on the underlying physical properties of the water, do not intuitively relate to the resulting visual appearance. We will present a novel approach which allows a range of video examples to be used as a set of visual parameters to design the visible behaviour of a water animation directly. Our work begins with a method for reconstructing the perceived water surface geometry from video footage of natural scenes, captured with only a single static camera. We show that this has not been accomplished before, because previous approaches use sophisticated capturing systems which are limited to a laboratory environment. We will also present an approach for reconstructing the water surface velocities which are consistent with the reconstructed geometry. We then present a method of using these water surface reconstructions as building blocks which can be seamlessly combined to create novel water surface animations. We are also able to extract foam textures from the videos, which can be applied to the water surfaces to enhance their visual appearance. The surfaces we produce can be shaped and curved to fit within a user's three-dimensional scene, and the movement of external objects can be driven by the velocity fields. We present a range of results which show that our method can plausibly emulate a wide range of real-world scenes, different from those from which the water characteristics were captured. As the animations we create are fully three-dimensional, they can be rendered from any viewpoint, in any rendering style.

Acknowledgements

Above all I would like to thank my supervisors Phil Willis and Peter Hall for their guidance and support throughout my PhD. Secondly I would like to thank Chuan Li for both his technical contributions to Chapter 3, but also for his advice and support throughout the rest of my project. My thanks also goes to Darren Cosker, Yi-Zhe Song, Andrew Chinnery and Matthew Brown, who have all offered useful advice and inspiration towards my work, and some of whom have allowed me to contribute to their own great research. I would also like to thank the University of Bath for funding my PhD, and Brian Wyvill and Mashhuda Glencross for examining my thesis. Finally I would like to thank my family who have given constant support during my studies.

Declaration of work done in conjunction with others

The work presented in Chapter 3 was done in conjunction with Chuan Li (University of Bath), who led the work presented in Section 3.4.

Chuan Li performed the work to capture the scanned surfaces in Section 3.5, along with Darren Cosker and Peter Hall from the University of Bath, and David Marshall from Cardiff University.

Publications

Conference proceedings

- Li, C., Shaw, M., Pickup, D., Cosker, D., Willis, P. and Hall, P., 2011. Realtime video based water surface approximation. In: 8th European Conference on Visual Media Production, CVMP 2011. Piscataway, NJ: IEEE, pp. 109-117.

- Pickup, D., Li, C., Cosker, D., Hall, P. and Willis, P., 2011. Reconstructing mass-conserved water surfaces using shape from shading and optical flow. In: Computer Vision, ACCV 2010 - 10th Asian Conference on Computer Vision, Revised Selected Papers. Vol. 6495 LNCS. Heidelberg: Springer-Verlag, pp. 189-201.

Journal articles

- Li, C., Pickup, D., Saunders, T., Cosker, D., Marshall, D., Hall, P. and Willis, P., Water surface modeling from a single viewpoint video. IEEE Transactions on Visualization and Computer Graphics 2012.

- Song, Y., Rosin, P., Li, C., Pickup, D., Hall, P., Abstract art by shape classification. IEEE Transactions on Visualization and Computer Graphics 2013.

Short papers

- Pickup, D., Cosker, D., Hall, P. and Willis, P., 2011. Statistically user controlled texture mixing. In: 8th European Conference on Visual Media Production, CVMP 2011.

List of Figures

Contents

Chapter 1

Introduction

In this thesis we shall present a novel method for example-based water animation. We will show that videos of real water scenes can be used as input examples from which novel three-dimensional scenes can be created. Our work allows a range of these video examples to be used as visual parameters to design the visible behaviour of a water animation. In this chapter we shall first introduce the objectives (Section 1.1) of our method, secondly we will describe the motivations to complete our objectives (Section 1.2), thirdly we will present the key challenges that we have had to face (Section 1.3), and finally give an overview of our main contributions (Section 1.4).

1.1 Objectives

The main objective of our research is to produce a practical and straightforward visual method of authoring novel three-dimensional animations of water from a set of example videos. As input we wish to be able to use example videos of natural scenes captured with an ordinary video camera. To be able to use these videos to produce three-dimensional scenes, we require a method of reconstructing the water surfaces from the video footage. To produce novel animations from a set of video examples, we require a method of combining video reconstructions within a novel scene environment. We want the authoring process of the user to involve specifying the visible behaviour of the animation directly, without the need to understand the underlying

method or the real-world physical properties of water. We require that our method be able to create novel water animations which plausibly mimic real-word scenes.

1.2 Motivations

The animation of water is an integral part of computer graphics as water is present in many real scenes, therefore there has been a large amount of research in this area. As we will show in Chapter 2 the work in this area has focused on creating physical simulations of water, which were originally devised by engineers and physicists studying fluid mechanics, requiring physically accurate results. They therefore take as input a set of parameters which relate to the underlying physical properties of fluids. To produce an animation the user is required to define the correct physical parameters which will result in the water behaving in such a way as to produce their desired visual effect. Such simulations are often very slow to run, and therefore the user sometimes has to wait a long time to see if the results fit their requirements. The user may then need to refine their choice of parameters if the simulation result is incorrect, leading to this expensive process being repeated.

For most computer graphics applications the physical accuracy of a water animation is unimportant, but the quality of the result is defined by its visual plausibility. This is because the animations are produced to be viewed by human beings. Producing physically accurate results, and producing visually plausible results are not identical problems because the human visual system can sometimes be fooled with physically inaccurate images. Work in other areas of computer graphics has taken advantage of this to develop simple approaches to creating effects that would otherwise be very difficult to achieve [44, 32, 30]. There is therefore strong justification for a method which instead of concentrating on the physical behaviour of a water animation, concentrates on the visual appearance. This would eliminate the need for the user to define physical parameters, which do not always intuitively relate to the visual result.

Video footage of real scenes provides good examples of the visual appearance of different water surfaces. There has been some work on using video to produce three-dimensional reconstructions (Chapter 2), but these methods do not work on natural scenes due to the requirement of a sophisticated

laboratory capture setup, in order to achieve a physically accurate reconstruction. There is a research gap for a method which uses a simple capture setup which is practical to use on natural outdoor scenes, concentrating on producing a visually plausible reconstruction of the water surface. The existing work is also split into methods which either reconstruct water surface geometry, or attempt to track the horizontal velocity of the water surface. Therefore, there is also motivation for a single method which produces both the surface's geometry and velocity which conform with one another.

If a selection of reconstructions of different water scenes could be combined seamlessly, the videos used as input to the reconstruction process could be used as a set of visual parameters when designing novel water animations. Using such parameters would allow the user to define the visual appearance of their desired animation directly, resulting in a more intuitive artistic process. In order for the user to be able to emulate large scale scenes, individual water surfaces would need to be extendible to fill an area of any size. Similarly to create long animations, surfaces reconstructed from videos of finite length would need to be extended or looped seamlessly in time. Using reconstructions which include velocity as well as geometry information, would allow the animations to contain external objects driven by the movement and flow of the water.

1.3 Challenges

There are several key challenges which have to be overcome to provide a solution to our objectives:

1. Reconstructing water surface geometry from ordinary video footage of natural scenes. Both the behaviour and optical properties of water are extremely complex.

2. To allow the water to interact with external objects within a novel scene. To do this a velocity field is needed, but water lacks distinctive visible features which can easily be tracked using existing computer vision techniques.

3. The velocity of the water and its surface geometry must be plausibly consistent with one another.

4. Combining different water surfaces together, with seamless transitions between different surface behaviours.

5. Creating large scale scenes and arbitrarily long animations from short video clips of water surfaces able to fit within a video frame.

6. Solving all the above challenges without the user having to define any complex physical parameters.

1.4 Contributions

The overall contribution of this research is a method of authoring novel three-dimensional animations of water scenes, from a library of video footage of real-world water surfaces. Our approach allows the user to directly define the desired visual behaviour of their required animation, without the need to understand the physical equations governing real-world fluid dynamics. The output animations are likely not physically accurate, but they fulfil the computer graphics requirement of visual plausibility. Our approach succeeds in meeting all the challenges presented in Section 1.3.

Our work consists of two major contributions which we outline below. Figure 1.1 shows an overview of our novel animation process.

1.4.1 Water Surface Reconstruction from Video

Our first major contribution is a novel method for reconstructing the water surface geometry and velocity from video footage of natural water scenes. This addresses Challenges 1, 2, 3 and 6. Previous methods for reconstructing the water surface geometry (Chapter 2) have concentrated on producing physically accurate reconstructions and have therefore not been able to reconstruct natural outdoor scenes. This is due to the sophisticated capturing systems they require, which only work in a laboratory environment. As the target application for our reconstruction approach is computer graphics, we require only a reconstruction which plausibly matches the user's visual perception of the shape of the water surface. This difference in objective allows us to reconstruct height fields of natural outdoor scenes using a simple and

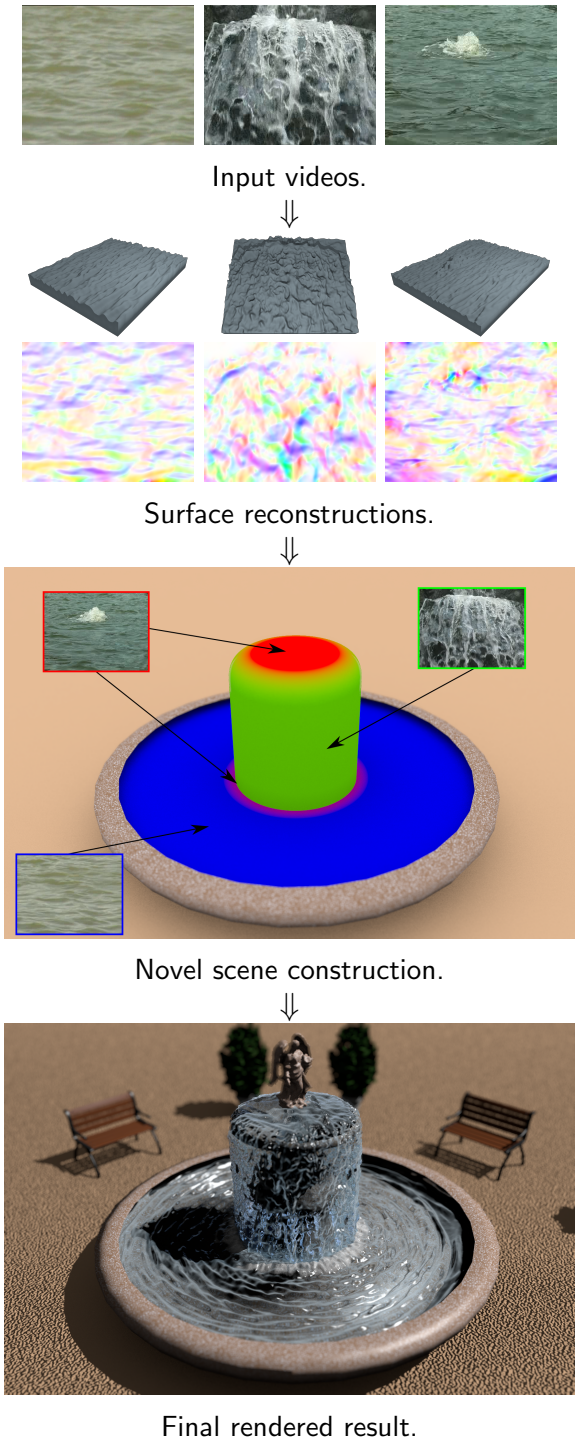


Figure 1.1: Overview of our novel animation process.

practical capturing system, consisting of a single static video camera. This solves Challenge 1.

Previous work either reconstructs the water surface geometry, or tracks its velocity. We require a coherence between the surface geometry and velocities and have therefore developed a novel tracking method which takes an animated water surface as input and produces a physically plausible velocity field. We achieve this using mass-conservation from the shallow water equations as a physical link between surface height and velocity. This solves Challenges 2 and 3. The physical link we use does not require the user to input any physical parameters to the reconstruction algorithm, therefore this solves Challenge 6 for our reconstruction approach.

1.4.2 Authoring Novel Animations of Water (Tiling)

Our second major contribution is a new approach to authoring novel animations of water using a library of video footage. This addresses Challenges 4, 5 and 6. The user is able to combine a range of water surfaces, reconstructed from different videos, into a single novel scene. The key to this is our approach of using water surfaces as building blocks which can be tiled together within a three-dimensional scene, with a seamless transition between the different water surface behaviours. The seamless transition is accomplished through our novel method of blending across an overlap region of two adjacent water surfaces. The inclusion of a velocity field in the resulting animations allows the movement of other objects within the scene to be driven by the movement of the water. These water surfaces can be shaped to match the geometry of a modelled scene, while preserving the correct surface motion and movement of objects driven by the velocity field. This work solves Challenge 4.

We have also developed an approach to allow our water surface tiling method to extend the size of a water surface reconstructed from a single video, allowing it to fill a larger area, while avoiding any visible spatial repetition. We are also able to use our tiling method to create a seamlessly looping animation, from an animated surface with a finite number of frames. This solves Challenge 5.

Foam is an important visual characteristic of natural scenes, and we are able to extract foam textures from video which can be applied to our water surface animations. This enhances the visual plausibility of the final scenes.

None of the methods within our authoring approach use any physical equations and therefore solve Challenge 6.

1.5 Thesis Structure

Here we have presented an introduction to our work. The remaining chapters of the thesis will be organised as follows:

- Chapter 2 will outline the main body of existing literature related to our research.
- Chapter 3 will present our new method for reconstructing water from video footage of real scenes.
- Chapter 4 will present our novel method for authoring animations of water from a selection of video reconstructions.
- Chapter 5 will present our conclusions.

Chapter 2

Background

The topic of water animation has been extensively researched in computer graphics, but that research has focused on the physical simulation of fluids. We therefore begin our discussion of previous work in this area of research (Section 2.1). We discuss how images and video have been used to aid in the modelling and animating of complex objects, focusing on fluids, in Section 2.2. In this section we discuss methods which produce both two-dimensional and three-dimensional animations. We then review in Section 2.3 previous work on the specific case of water reconstruction, as this area is most relevant to our work. Finally, because we later propose a method of tiling together water surfaces (Chapter 4), we review existing work in the related field of example-based texture synthesis in Section 2.4.

2.1 Water Simulation

The field of water animation has focused on developing methods of simulating the physical behaviour of real fluids. Our work is not simulation based. We provide a brief review of some of the main work in this area to provide a justification for an alternative solution. We also borrow some from the physical equations used by some of these methods in our reconstruction work (Chapter 3), so it is convenient to review them now.

Fournier and Reeves [29] proposed one of the first methods of simulating

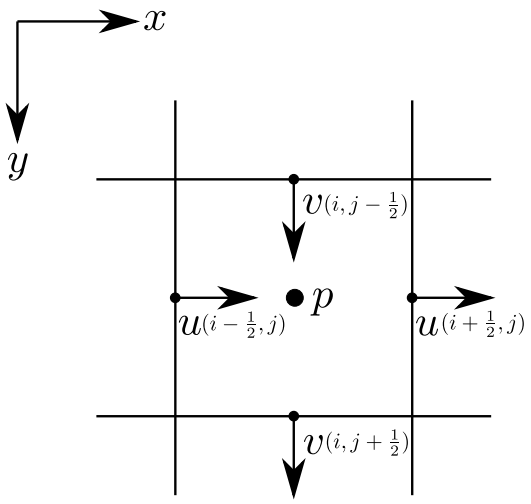


Figure 2.1: Marker-and-cell grid used in simulation.

water. They proposed a method of producing animations of an ocean shoreline using Gerstner waves. Peachey [67] developed a similar method which combines a combination of wave profiles. Mastin et al. [54] filter an FFT magnitude spectrum consisting of random noise using a modification of the Pierson-Moskowitz model of ocean waves [69]. The inverse-FFT of the result produces a height field of an ocean surface. They produce an animation by manipulating the FFT phase. Tessendorf [85] proposed a similar method, but using the Phillips spectrum to filter the FFT magnitude. These methods produce convincing results but are all limited to producing ocean surfaces.

A common method of animating fluid is to simulate the Navier-Stokes equations:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \tag{2.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2.2}$$

where $\vec{u}$ is a 3D velocity vector (u, v, w), p denotes pressure, ρ density, ν viscosity, and g forces acting upon the fluid [11]. The advection of fluid through space is modelled by Equation 2.1 and its incompressibility by Equation 2.2.

Foster and Metaxas [28] proposed a grid-based method for solving the Navier-Stokes equations. Their method represents the simulation volume as a marker-and-cell (MAC) staggered grid [33], which stores a pressure term for each cell and a velocity component at the centre of the cell faces (Figure 2.1). At each time-step of their algorithm they update the velocities and pressure for each grid cell individually, using the Navier-Stokes equations. The key weakness

of their algorithm is that it becomes numerically unstable for large time-steps. Stam [82] extended this method by introducing the semi-Lagrangian method for solving advection. To calculate the fluid velocity for a point x at a new time-step $t + \Delta t$, this algorithm backtraces the point x through the velocity field over a time Δt. The velocity at x for time $t + \Delta t$ is then set to the velocity of this new position at time t. This algorithm is unconditionally stable, which allows for large time-steps for increased efficiency. Level set methods have been developed to allow the animation of complex liquid surfaces using grid-based methods [27, 24, 23]. These methods use a combination of weightless marker particles and an isocontour of an implicit function to model the liquid's surface.

As an alternative to grid-based methods, Müller et al. [59] proposed a particle-based method based on Smoothed Particles Hydrodynamics [57] to simulate liquids. Their method derives the forces acting upon each particle individually using the Navier-Stokes equations. Particle simulations can better simulate the finer details of liquid movement, and the particles can be directly used to render the surface of the liquid.

Simulating the full Navier-Stokes equations is computationally expensive. Methods which use the shallow water equations to simulate water have been developed to avoid this expense. These produce a time-varying height field as the geometry of the water surface. Although these methods are more efficient than Navier-Stokes simulations, this prevents them from producing effects such as splashes and breaking waves. Kass and Miller [43] approximate the shallow water equations using a version of the wave equation. Their method suffers from numerical instability for large time-steps. Layton and van de Panne [48] proposed an unconditionally stable method using the full shallow water equations and the semi-Lagrangian advection method proposed by Stam [82]. Chentanez and Müller [13] also simulate the shallow water equations. Their method is not unconditionally stable, but they include stability enhancements which clamp the values of certain simulation variables. Methods which use particles to enhance the range of effects produced by shallow water simulations have been developed. These additional effects include breaking waves [89, 13], splashes [13], and bubbles [88].

There also exist methods of coupling different types of simulation [90, 13] or simulations of different resolutions [81, 14]. The aim of these methods is to have a highly detailed level of simulation surrounding an area of interest, and a lower-cost simulation filling the rest of the scene. All these methods perform the coupling on the actual physical simulations, not on the resulting

water surface geometry. Wicke et al. [97] proposed a method of designing a scene by tiling together precomputed modular simulation tiles, which are reduced models created from high-resolution simulations. This work allows the simulations to fill larger domains than standard techniques. Their method is designed to simulate wind, not water.

Fluid simulations are able to produce convincing animations, but are difficult for artists to control. The parameters of a simulation (e.g. density, viscosity, external forces) are physically based, and do not intuitively relate to the visual appearance of the resulting animation. Several methods of aiding artists more easily to control fluid simulations have been developed. Here we concentrate on previous work of controlling the simulation of liquids.

Foster and Metaxas [26] were the first to propose such a control mechanism. They proposed placing embedded controllers within the simulation to allow the animator to control the pressure and velocity of the local fluid flow, to achieve effects such as fountains and explosions. Foster and Fedkiw [27] extend this work to allow the velocity of the fluid to be controlled by sampling three-dimensional parametric space curves with oriented points.

Rasmussen et al. [72] proposed a method of directing a liquid by placing throughout the fluid volume control particles which can control a variety of liquid variables, including the isosurface value, velocity and viscosity. This technique was designed to produce the melting effect of the female terminatrix in the film Terminator 3. The method requires much manual input by the user, as well as an understanding of the physical simulation parameters.

Several methods have been produced which allow an artist to fit an animated liquid to a target shape. Treuille et al. [92] presented an optimisation method to constrain smoke to conform to user defined density and velocity keyframes. It achieves this by solving for appropriate forces to be applied to a fluid simulation. McNamara et al. [56] greatly improve the efficiency of this method by using the adjoint method to solve the nonlinear optimisation problem. With this improvement the technique can be used to control liquid simulations as well as smoke. Even with the efficiency enhancements, this method is too computationally expensive to use with the high resolution grids commonly used for liquid animations within the computer graphics industry [72]. Their results include a water simulation with a grid resolution of 45x50x36, controlled to appear as a "running man", which took two days to compute. Shi and Yu [80] control a liquid by matching its level set surface to a static or moving target shape. They apply two external force fields to

the simulation: a feedback force field to compensate for discrepancies in both shape and velocity, and the gradient field of a nonlinear potential function defined by the shape and skeleton of the target object. Thürey et al. [86, 87] control the shape and velocity of a liquid using control particles generated automatically from either a previous physical simulation or a sequence of target shapes. The control particles apply both an attraction and velocity force to the liquid. The control forces only affect the larger liquid motion, which preserves the small scale details. These methods are good at controlling the global shape of a water volume, and often demonstrate their ability to produce effects such as a liquid spelling words or forming liquid characters. They are not aimed at controlling the finer water movement, such as the waves moving across a water's surface, and therefore are less useful when mimicking natural scenes.

Existing control mechanisms either require the user to understand the simulation framework, or they only control the global shape of the liquid and not the finer wave movement. Some of these methods also add an extra computational cost to an already expensive method of animating liquids.

2.2 Video-Based Graphics

Images and video have been used to aid the modelling of many different complex objects [10], which includes trees [84, 50], hair [66], fabrics [99], human bodies [83], faces [8], plants [71], buildings [62] and fluids. As our work concerns water, the previous work on fluids is particularly relevant. The reconstruction of fire has been proposed by Hasinoff and Kutulakos [34, 35] who represent the fire as a density field computed from the convex combination of sheet-like density fields derived from pairs of input images. Ihrke and Magnor [41] proposed a tomographic method of reconstructing fire from images captured with a ring of eight cameras. This method was later extended to reconstruct a volumetric model of smoke [42]. Hawkins et al. [36] also reconstruct smoke animations, but using a different capture setup consisting of a laser and high speed camera. Atcheson et al. [4] use a Schlieren tomography system to capture non-stationary gas flows, such as hot air, on a dense volumetric grid. Dobashi et al. [18] synthesise a density distribution of clouds, which is similar to clouds visible in a single example photograph. Several methods of reconstructing water from video footage have been proposed, and we dedicate the next section to this area of work as

it is more closely related to ours (Section 2.3).

The previous methods of reconstructing three-dimensional fluids, including water, from video do not attempt to use the output reconstructions to produce novel animations not fully described by the original input video. This is one of the main research gaps left by previous work. However producing novel water animations from video has been explored in two-dimensions. Bhat et al. [9] synthesise videos of fluid by first capturing the motion and texture variation along a set of user-defined flow lines drawn over a fluid in a video. The user can then specify a new set of flow lines, and the same motion and texture variation are applied to these to produce a novel two-dimensional animation. Their method does not address the issue of combining motion from multiple examples. Okabe et al. [64, 65] transform a single image of a fluid into a video, using a fluid video database. They accomplish this by automatically replacing static patches in the original image with video patches that have a similar appearance. Similar to our work, they have a method to blend over the boundaries of adjacent patches. Their method is more complex than ours, as it blends videos directly, and relies on information from the original target image. Their use of a target image also prevents them from producing truly novel scenes.

Methods to transform static images of fluids into videos without the use of video examples have also been developed. Chuang et al. [15] animate water in a still image by simulating a height-field animation, using the spectrum-based method by Tessendorf [85], on a three-dimensional water plane calculated from the image. They then map the heights on the water plane to displacements in image space. Lin et al. [53] generate a video of a fluid from a collection of still images by ordering the images to match the scene dynamics, and feathering between frames. Sakaino et al. [77] animate an image by moving pixels according to a user defined 3D velocity field. These methods are limited to producing a scene depicted by a single real-world image, and can therefore not be used to design novel scenes.

2.3 Water Reconstruction

In Chapter 3 we propose a new method of reconstructing both a time-varying height and velocity field representation of a water surface from video. Our work aims to provide a practical solution to creating novel water animations,

using video as input examples. In order for our method to be practical for a wide range of scenes and easy to use for a non-technical user, the video footage we use is of natural outdoor scenes and captured with a standard video camera. Here we review the previous work on reconstructing water from video. These methods fall into two groups: surface geometry reconstruction, and velocity tracking. We review these two groups separately.

2.3.1 Surface Geometry Reconstruction

Here we give an outline of the previous work in water geometry reconstruction.

Several methods of reconstructing water surfaces have been proposed. Murase [61] tracks the distortion of an underwater pattern with optical flow, and the surface's normal is calculated using a refraction model. Morris and Kutulakos [58] reconstruct the water surface by minimising refractive disparity, assuming light is refracted only once. Balschback et al. [6] also use a refraction approach, but based on a shape from shading technique, where multiple illuminations, both above and below the water's surface, are used to better determine surface gradients. This group of methods require controlled conditions, most easily contrived in a lab. They are not suitable for capture from natural outdoor scenes where there is often no visible distinctive pattern or source of illumination visible below the water surface.

Ihrke et al. [40, 31] measure the thickness of water dyed with Fluorescein from the amplitude of the emitted light, which is used as a constraint when calculating the visual hull of the water surface. Shape from stereo techniques have also been explored. Wang et al. [94] dye water with white paint, project a light pattern onto its surface and capture it through a pair of video cameras. They refine their reconstruction using physically-based constraints. Hilsenstein [38] reconstructs water waves from thermographic image sequences acquired from a pair of infrared cameras. These methods produce high quality results but require sophisticated equipment and complex experimental setups, which again limits their application to a laboratory setting.

2.3.2 Velocity Tracking

As the aim of our work is to use videos to produce visually realistic novel water animations, we wish to capture the water's velocity, in addition to its geometry, to allow objects to be placed on the water and plausibly move with the water's flow. Here we give an overview of previous work on tracking the velocity of water from video.

Traditional video tracking algorithms such as optical flow [5] typically use the constant brightness constraint. This assumes that as a pixel moves from one location to another between images, its intensity does not change. Standard optical flow methods typically don't perform well for tracking water, as water exhibits few distinct visual features which can be tracked by the brightness constraint. As an improvement, Nakajima et al. [63] extend the Horn-Schunck optical flow method [39] by adding a constraint based on the incompressibility condition and momentum equation from the Navier-Stokes equations. They simplify the problem by limiting the Navier-Stokes equations to two-dimensions. Doshi and Bors [19] use a robust kernel which adapts to the local data geometry in the diffusion stage of the Navier-Stokes formulation. The kernel ensures that smoothing occurs along the structure on the motion field whilst maintaining the general optical flow structure and the main optical flow features. They claim to track fluids in general, but demonstrate their method only on videos of atmospheric and solar phenomena. Sakaino [76] proposes a method to model abrupt image flow change. Flow is modelled using a number of base waves and their coefficients are chosen to match the input sequence. Li et al. [51] treat an image as a wavefront surface and derive a general brightness constraint to model brightness variation in terms of fluid dynamics of the velocity potential flow.

2.4 Texture Synthesis

Some of the aims of texture synthesis are superficially similar to our work on combining animated water surfaces. This is because water surfaces can be interpreted as "dynamic textures". We therefore outline the key works in texture synthesis. In this section, when we use the term 'texture image' we are referring to an image which displays a visually repeating pattern. Four examples of texture images are shown in Figure 2.2. There are many

Figure 2.2: Four examples of image textures.

different ways to synthesise texture images, but the one which is relevant to our research is texture synthesis from example. The aim of the work in this area is to observe an example texture, and synthesise a new image which appears to depict the same texture as the input example, but is unique. Often texture synthesis is used to make a texture image larger than the input example. Methods for producing textures from multiple examples, and methods for synthesising video textures from example have also been proposed. Here we will give an outline of some of the major work in texture synthesis, in order to demonstrate their capabilities and show where they fall short of providing us with a solution for our work. For a more detailed review we refer readers to [95].

Heeger and Bergen [37] use a parametric method to synthesise textures. Their algorithm accepts an example texture and a random noise image as input, and matches the histograms of these two images. A steerable pyramid is then calculated from each image and histogram matching is then performed at each layer of the pyramid. The synthesised texture is then created by collapsing the noise pyramid. De Bonet [17] also proposed a multi-resolution approach to texture synthesis. This method first builds a multi-resolution pyramid from an example texture. The algorithm then creates the synthesised texture by sampling from each layer of the example pyramid in order starting from the lowest resolution level. The sampling at each level of the synthesis pyramid is constrained by the local parent structure. Bar-Joseph et al. [7] extend this work using wavelets instead of an image pyramid. Their method is able to mix together multiple static textures, and also synthesise video textures using a three-dimensional wavelet transform.

A more common trend within texture synthesis is to model a texture as a Markov Random Field, where each pixel is characterised by its local neighbourhood and each neighbourhood of pixels appears similar. This implies that each pixel can be predicted from observing a small set of neighbouring

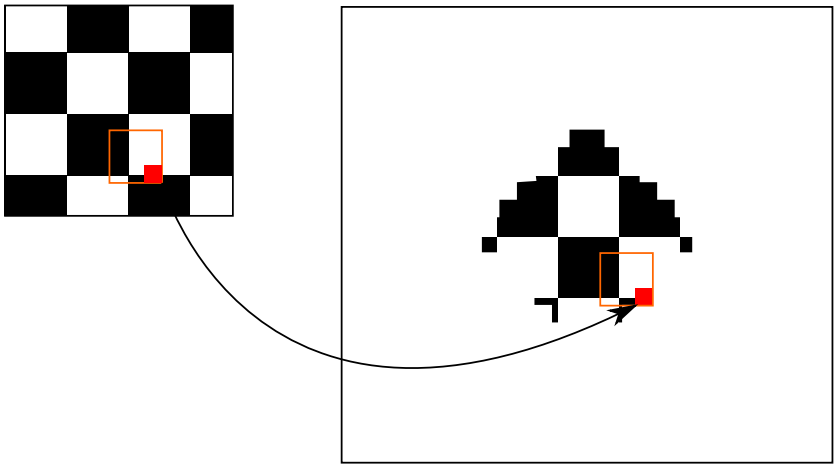


Figure 2.3: The red pixel on the synthesised image (right) is copied from the example image (left) based on the similarity of their local neighbourhoods.

pixels. Efros and Leung [21] use the Markov Random Field model to grow a texture inside-out from an initial seed, one pixel at a time from an example image. Their algorithm is illustrated in Figure 2.3. For each pixel in the synthesised texture, its already synthesised local neighbourhood is compared against all possible neighbourhoods of the same size in the example image and a set of best matches is produced. One of these best neighbourhoods is chosen at random, and the pixel in the same relative position to the neighbourhood as the pixel to be synthesised is copied to the synthesised image. A similar technique was proposed by Wei and Levoy [96]. The difference with their method is that they use a fixed neighbourhood size for each pixel that is synthesised and they can then perform the synthesis in a scanline ordering starting from the top-left corner of the image. They are able to use a fixed neighbourhood size by first initialising the synthesised image to random noise before performing their algorithm. They also use their method in three-dimensions to synthesise video textures. Ashikhmin [3] extends this work and allows the user to produce a colour map to guide the placing of texture elements in the output image. When synthesising a pixel in the output image, the algorithm favours copying pixels of a similar colour to the corresponding colour on the colour map.

Efros and Freeman [22] extended this area of work by developing their image quilting algorithm. Instead of pixel by pixel, this method synthesises a new image by tiling together patches copied from a sample texture, where the boundary of each pair of adjacent tiles matches in appearance. Each pair of adjacent patches also overlap slightly and a lowest error seam is calculated

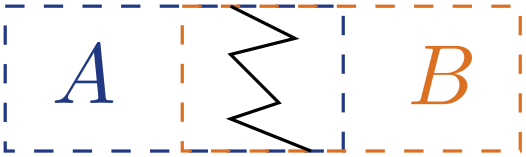


Figure 2.4: The lowest error seam between two overlapping texture patches.

using dynamic programming (Figure 2.4). Liang et al. [52] proposed a similar algorithm to image quilting, but instead of calculating an optimal seam they simply blur the patch boundaries which causes more obvious visual artifacts.

Kwatra et al. [46] extended image quilting by formulating a graph from the overlap region between two patches, and using a graphcut algorithm to calculate the optimal seam. The advantage of using a graphcut algorithm is that it can be performed in three dimensions, whereas dynamic programming is limited to two. This allows the algorithm to synthesise video, as well as static images. This graphcut technique has been extended by Agarwala et al. [2] to produce a video panorama from video footage of a scene captured by a panning video camera.

Cohen et al. [16] developed a method of producing a set of Wang tiles from an example texture. Each Wang tile has a colour assigned to each of its edges, where adjacent tiles which have the same coloured edge on the side they contact have a seamless boundary. An output image can therefore be generated by tiling the Wang tiles over the image, with the constraint that each adjacent pair of tiles share the same edge colour where they are aligned. Lagae and Dutré [47] propose using coloured corners instead of edges. Chenney [12] generate a set of static vector fields, called flow tiles, which can be tiled together to produce a divergence free flow if similar boundary conditions are adhered to.

Schödl et al. [79] focus solely on video textures, rather than image textures. They are able to produce an infinitely long video texture by finding sets of matching frames. They either transition between these frames during playback, or they crop the video at two matching frames to produce an infinitely repeating video with a seamless loop.

Sometimes the aim of texture synthesis is not only to match the characteristics of a single input example, but to mix the characteristics of two or more examples. This not only allows novel textures to be designed, but it sometimes allows a texture to morph from one to another across the space of an

image. An extension to the Wang tile method [16] has been proposed by Schlömer and Deussen [78] which allows the user to specify the probability for the placement of each edge colour at each position within the output image. The user therefore has an increased level of artistic control over the appearance of the output. They also allow tiles to be generated from multiple example textures to allow for texture mixing. We proposed a similar method [68], where we extend image quilting [22] by allowing the algorithm to select each patch from one of a selection of example textures. This choice is made based on a user specified probability map which varies over the target image.

Matusik et al. [55] proposed a method of designing a texture from multiple examples, by actually blending between them. First a database of example textures is put into a high dimensional space and linked together into a simplicial complex, where vertices represent the input textures. Two vertices are only connected if the distance between them obeys an adaptive distance criterion. Pairs of connected textures can then be morphed together using their colour and a warping function which aligns their features, which are extracted using the method by Ruzon and Tomasi [75].

A material space representation of textures was proposed by Ray et al. [73]. Their method represents a texture coordinate as (s, t, m), where s and t denote the spacial coordinates and m represents the weighted warp between two different textures. The warping function they propose uses advection to interpolate between two level sets representing the two different material feature masks. Instead of calculating a global warping function, Ruiters et al. [74] proposed warping smaller image neighbourhoods prior to patch-based texture synthesis. Once this is complete, a single iteration of the statistical texture synthesis algorithm by Portilla and Simoncelli [70] is used to synthesise the high frequency details otherwise lost during interpolation.

2.5 Conclusion

The previous work we have discussed leaves open some problems which we address in the rest of this thesis. The work on fluid simulation arose from the need for physical accuracy, and as a result simulations are computationally expensive and the visual result is unintuitive to define. This is because they take as input physical parameters, such as density and viscosity. The work which addresses the controllability of fluid simulations concentrates on the

global behaviour of the fluid, for creating unnatural effects such as characters made of fluid. It does not aid in defining behaviours such as the shape and speed of waves rippling along a water surface, and is therefore less useful when controlling the appearance of natural animations, such as rivers and lakes. The prior work in this area confirms that there is a clear gap for a method which allows the user to design water animations using visual parameters, instead of physical parameters, which allows the user to intuitively produce animations which mimic real-world scenes. Our proposed approach will fill this gap.

The methods we have described which reconstruct water surface geometry aim to create physically accurate reconstructions and therefore all require sophisticated capture systems which limits their use to a laboratory environment. These methods can not be used to reconstruct natural outdoor scenes. We will show how we address this issue, using a simple capture setup designed for outdoor scenes (Chapter 3).

The previous work on reconstructing water either reconstruct the surface geometry, or a velocity field. There is room for a single method which reconstructs both the geometry and the velocity which are coherent with one another. We will address this in Chapter 3.

Some of these methods for texture synthesis could be applied to our problem of extending the size of one of our water surfaces, or producing an animation which can be looped seamlessly. The problem they can not solve is combining multiple surfaces together to produce a novel animation. We have described methods which are able to blend from one image texture to another, but these rely on the static structure of the texture and therefore can't be applied to animated textures. We address this in Chapter 4.

Chapter 3

Water Surface Reconstruction from Video

The aim of our work is to create novel three-dimensional animations of water from a collection of video footage. In order to achieve this the first thing we need is a method of reconstructing a water surface from video. We simplify the problem to reconstructing a height field representation of the water surface geometry, which plausibly matches the user's visual perception of the shape and movement of the water surface visible in the video. This allows us to reconstruct a water surface from video footage of natural outdoor scenes, such as the videos shown in Figure 3.1, without the need of a complex experimental setup.

As well as producing an animated height field, we also produce a dense time-varying two-dimensional velocity field of the horizontal movement of the water. The velocity field is needed when we later create novel water animations which contain other objects that we want to move with the movement of the water (Chapter 4). As detailed in section 2.3, existing reconstruction methods either reconstruct the water's geometry or its velocity, but not both. In order to fulfil our aim to produce highly plausible animations, we require that the geometry and velocities we reconstruct are coherent with one another. We achieve this coherence by developing a single method which produces both the geometry and velocity using a physically based link between them.

The use of height fields allows us to represent a wide range of surface effects. We will demonstrate that whilst using this representation our method is still

Figure 3.1: Several examples of our input videos.

able to reconstruct a visually convincing approximation of some non height field phenomena, including waterfalls, breaking waves, splashing and boiling water.

Our method uses a mass-conservation term, taken from the shallow water equations, as a function describing the link between the change of the surface height and the horizontal velocities. Shape-from-shading is first used to acquire a time-varying height field representation of the surface geometry. The change of surface height over time is then used, in conjunction with mass-conservation, as a prior to constrain an optical flow tracking algorithm which produces a time-varying velocity field. The final surface geometry is then recalculated from these velocities, using the same mass-conservation term.

The rest of this chapter will first introduce the mass-conservation term, then demonstrate how shape-from-shading is used to acquire the water surface geometry, and thirdly describe the constrained velocity tracking. Since this work was originally completed, work to extend it has been led by a colleague. We shall discuss this extension separately to distinguish it from work led by the author. We shall then present the results of our method, and finally conclude.

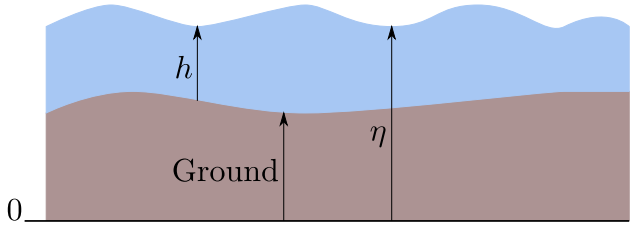


Figure 3.2: The height values in the shallow water equations. η is the height above zero, and h is the height above ground level. If the ground lies at zero then h is equal to η.

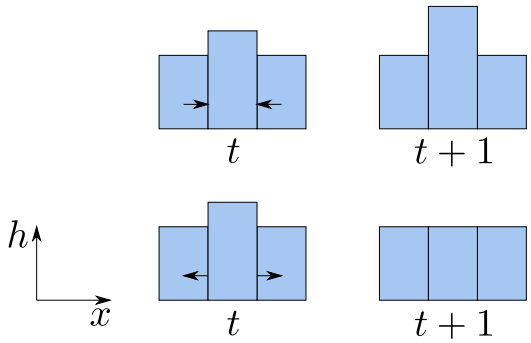


Figure 3.3: The incompressibility condition ensures conservation of mass. Top: If more water enters a region in the horizontal plane, the height of the water increases. Bottom: If more water leaves a region the height of the water decreases. The arrows represent the fluid velocity.

3.1 Conservation of Mass

As well as reconstructing the water surface geometry, we also require a velocity field reconstruction to accomplish some of the work reported in Chapter 4. For our results to appear plausible, we require a coherence between the surface geometry and velocity. To achieve this coherence we use a function describing the relationship between the height and velocity fields when reconstructing them from video. We derive this function from the shallow water equations, which assume the water surface can be represented as a height field, and that the height of the waves is small compared to their horizontal scale.

A basic version of the shallow water equations often used in computer graph-

ics [60] can be written as

$$\frac{\partial \eta}{\partial t} + (\nabla \eta)\vec{u} = -\eta \nabla \cdot \vec{u}, \tag{3.1}$$

$$\frac{\partial \vec{u}}{\partial t} + (\nabla \vec{u})\vec{u} = g\nabla h, \tag{3.2}$$

where h is the height of the water above zero, η is the height of the water above ground level (Figure 3.2), $\vec{u}$ is the water's horizontal velocity (u, v), and g is vertical acceleration due to gravity (-9.81m/s^2). We immediately simplify these equations by assuming the ground level is zero everywhere. This eliminates the term η, replacing all its occurrences with h. Water simulations are sometimes simplified by ignoring advection [60]. Taking the same approach, we can simplify our optical flow process (see Section 3.3). This gives the following equations

$$\frac{\partial h}{\partial t} = -h\nabla \cdot \vec{u}, \tag{3.3}$$

$$\frac{\partial \vec{u}}{\partial t} = g\nabla h. \tag{3.4}$$

Equation 3.3 is identical to the Navier-Stokes incompressibility condition

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{3.5}$$

assuming the gradient of the vertical velocity w with respect to z is defined as

$$\frac{\partial w}{\partial z} = \frac{1}{h}\frac{\partial h}{\partial t}, \tag{3.6}$$

where x and y are the horizontal axes, and z is the vertical axis. The incompressibility condition is what ensures the conservation of mass, i.e. that if some fluid enters a region the same quantity of fluid must also leave the region. In our case if the horizontal velocity field shows more water entering a region than leaving it, the height of the water in that region must be increasing. In the opposite case where the velocities show more water exiting a region, then the height must be decreasing. This is illustrated in Figure 3.3.

Our reconstruction method consists of first obtaining a height field for every video frame using shape-from-shading. We then wish to calculate a surface velocity field from each adjacent pair of frames using an optical flow algorithm. From the incompressibility condition (Equation 3.5) we can use our

reconstructed height field to estimate the required divergence of the velocity field as follows

$$\nabla \cdot \vec{u} = -\frac{h(x, y, t+1) - h(x, y, t)}{h(x, y, t)}. \tag{3.7}$$

This incompressibility condition is therefore used as our function describing the relationship between the height and velocity fields.

3.2 Recovering Surface Geometry using Shape-from-Shading

The first stage to our approach is to reconstruct the water surface geometry in the form of a time-varying height field. A height field is simply the inverse of a depth map, and a common method for reconstructing depth from a single image is shape-from-shading. Shape-from-shading deals with the recovery of shape from a gradual variation of shading in an image, see Zhang et al. [98] and Durou et al. [20] for detailed surveys of various methods. A typical assumption made by shape-from-shading techniques is that the scene follows the Lambertian model, in which the grey level at a pixel in the image depends on the light source direction and the surface normal.

Water has highly reflective and refractive material properties, and therefore invalidates the Lambertian assumption made by shape-from-shading. However, our experiments show that we are able to produce visually plausible results in practice (Section 3.6). It is unlikely that this is because the results we obtain are a physically accurate representation of the surface heights, but this is not our aim. Our aim is to produce a height field which, when rendered, produces an animation which is visually consistent with the perceived water shape and behaviour visible within the original video.

We use the method proposed by Tsai et al. [93] to reconstruct our height fields from video. We chose this algorithm as the source code is available, it reconstructs each of the videos we use very quickly, and produces visually plausible results. We do not make any changes to the original algorithm, and therefore we give a description of its workings in Appendix A.

To produce a reconstruction of one of our videos we first remove noise by applying a low-pass filter to each video frame. We then process each frame independently with the shape-from-shading algorithm. The result is a height

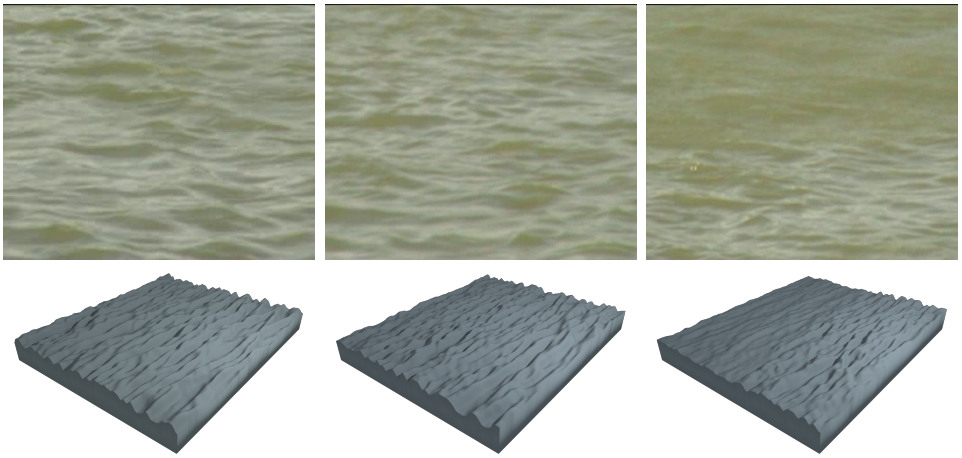


Figure 3.4: The shape-from-shading surface for several frames of the same video footage. Top: Video frames. Bottom: Shape-from-shading reconstructions.

$h(x, y, t)$, which lies in the range $[0, 1]$, for every (x, y) pixel position of every frame t, for each video. Several reconstructed frames for a single video are shown in Figure 3.4. We perform this procedure on each of the videos in our dataset.

The surfaces produced by shape-from-shading sometimes contain vertical drifts caused by global luminance change between frames. We rectify this by normalising the average height of each frame to zero

$$h'(x, y, t) = h(x, y, t) - \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} h(i, j, t), \tag{3.8}$$

where M and N are the number of pixels in x and y respectively.

Some of the videos we use have spatial luminance changes caused by shadows or reflections visible on the water surface. These produce unwanted dips and slopes on the resulting shape-from-shading reconstruction. We similarly correct for this by altering the height field so that the average height over time at each position (x, y) is equal to zero

$$h'(x, y, t) = h(x, y, t) - \frac{1}{T} \sum_{i=1}^{T} h(x, y, i), \tag{3.9}$$

where T is the total number of frames. This rectification makes the assumption that the water at each point is moving about the same average height

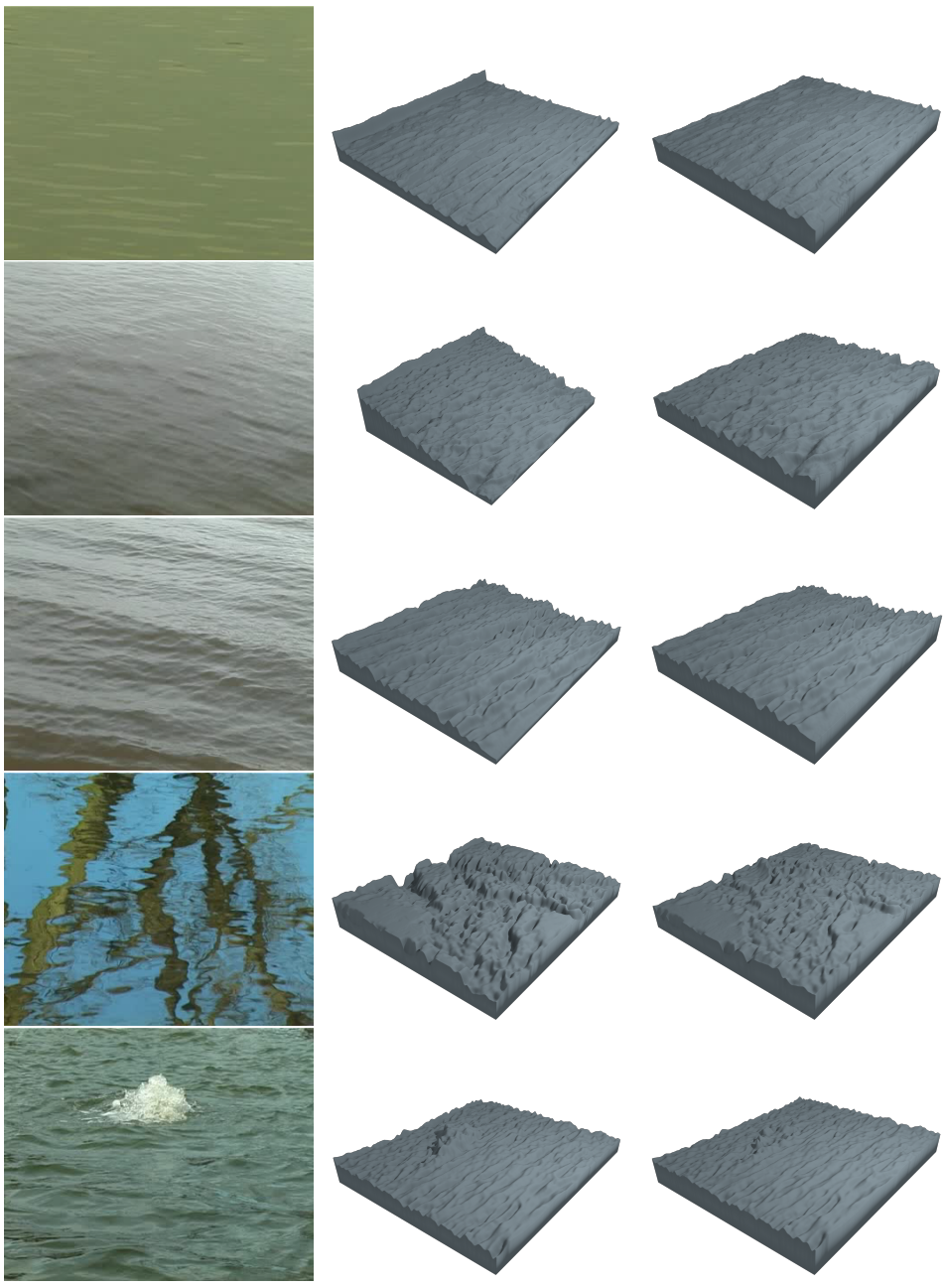


Figure 3.5: We correct dips and slopes caused by shadows and reflections. The fourth example shows reflections which create too much distortion for us to remove. The last example contains desired features that would be removed using our correction method. We therefore do not apply the correction to this type of surface. The surface heights are exaggerated for clarity of viewing.

[61]. Some of the videos we use do not conform to this assumption, and we therefore leave out this last step. Examples of both kinds are shown in Figure 3.5. Please note that as shape-from-shading does not determine the scale of the reconstructed scenes, we scale the surfaces appropriately for visualisation. Sometimes we exaggerate this scale so the surface shape can be seen more clearly.

The height fields are converted to surface geometry for rendering by placing a mesh vertex at each grid position of the height field and triangulating using Delaunay triangulation.

3.3 Tracking Surface Velocities using Constrained Optical Flow

At this point we have an estimate of the height for each video frame, acquired using shape-from-shading. We also require a velocity field which is coherent with the height field reconstruction. In Section 3.1 we described a function linking the height and velocity of a water surface, based on the conservation of mass. This function allows us to use a height field we have reconstructed as a prior to constrain a tracking algorithm. The algorithm we have chosen is the optical flow algorithm by Horn and Schunck [39]. This algorithm is computationally efficient, can easily be extended by adding extra constraints, and fulfils our requirement of producing a dense velocity field.

The original Horn and Schunck optical flow algorithm [39] calculates the velocity vector (u, v) which minimises an error made up of the sum of the squares of two constraints. The first is the brightness constraint, which works on the assumption that as an object moves between two frames its brightness is constant. This constraint therefore aims to minimise the rate of change of image brightness along the velocity vector (u, v)

$$\xi_b = E_x u + E_y v + E_t, \tag{3.10}$$

where E_x, E_y and E_t are the derivatives of image brightness E in x, y and t respectively. The estimates of the derivatives are calculated using a set of formulae which give an estimate at a point in the centre of a cube formed by

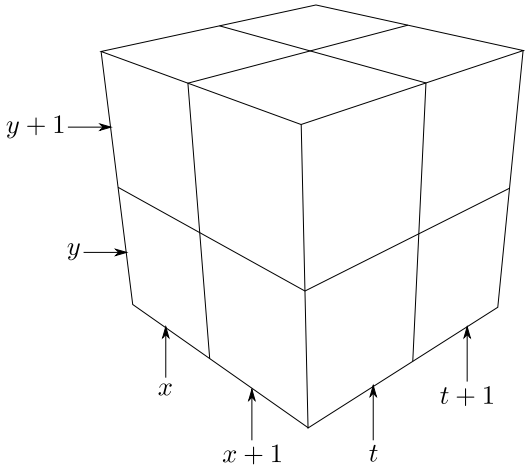


Figure 3.6: Each of the estimates for the derivatives of image brightness at the centre of the cube are the average of first differences along four parallel edges of the cube.

eight measurements (Figure 3.6). Hence

$$E_x = \tfrac{1}{4}(E(x+1,y,t) - E(x,y,t) + E(x+1,y+1,t) - E(x,y+1,t) + E(x+1,y,t+1) - E(x,y,t+1) + E(x+1,y+1,t+1) - E(x,y+1,t+1)), \tag{3.11}$$

$$E_y = \tfrac{1}{4}(E(x,y+1,t) - E(x,y,t) + E(x+1,y+1,t) - E(x+1,y,t) + E(x,y+1,t+1) - E(x,y,t+1) + E(x+1,y+1,t+1) - E(x+1,y,t+1)), \tag{3.12}$$

$$E_t = \tfrac{1}{4}(E(x,y,t+1) - E(x,y,t) + E(x+1,y,t+1) - E(x+1,y,t) + E(x,y+1,t+1) - E(x,y+1,t) + E(x+1,y+1,t+1) - E(x+1,y+1,t)). \tag{3.13}$$

The second constraint used by the Horn-Schunck algorithm is the smoothness constraint, which assumes that neighbouring points in the image will have similar velocities, therefore producing a smooth velocity field. This constraint is fulfilled by minimising the square of the magnitude of the velocity gradient

$$\xi_c^2 = (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2. \tag{3.14}$$

These two constraints are combined to produce the total error to be minimised

$$\xi^2 = \int\int (\alpha^2 \xi_c^2 + \xi_b^2) \mathrm{d}x\mathrm{d}y, \tag{3.15}$$

where α is a user defined weighting for the smoothness constraint.

We will later show that minimising the original Horn-Schunck error alone does not produce a satisfactory velocity field for water, as the water surfaces exhibit few visual features that can be easily tracked by the brightness constraint. Since we know that we are specifically tracking water, and we also have geometry data, we can use this extra information to improve the tracking performance. We do this by adding an additional constraint to the error, based on the conservation of mass described in Section 3.1

$$\xi_d = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}, \tag{3.16}$$

where the partial derivative of the vertical velocity w is calculated, using Equation 3.6, from the shape-from-shading result as

$$\frac{\partial w}{\partial z} = \frac{h(x, y, t+1) - h(x, y, t)}{h(x, y, t)}. \tag{3.17}$$

After including our additional constraint, the updated error to be minimised is

$$\xi^2 = \int\int (\alpha^2 \xi_c^2 + \beta^2 \xi_d^2 + \xi_b^2) \mathrm{d}x\mathrm{d}y, \tag{3.18}$$

where β is a user defined weighting for the mass conservation constraint. Both α and β are set to one in all our examples.

The optical flow track is calculated by finding the values for u and v which minimise the final error. This is done by solving the associated Euler-Lagrange equations

$$\frac{\partial L}{\partial u} - \frac{\partial}{\partial x}\frac{\partial L}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial u_y} = 0, \tag{3.19}$$

$$\frac{\partial L}{\partial v} - \frac{\partial}{\partial x}\frac{\partial L}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial v_y} = 0, \tag{3.20}$$

where L is the integrand of the error functional and u_x, u_y, v_x ,and v_y are the derivatives of u and v with respect to x and y respectively. All the terms in

1/12	1/6	1/12
1/6	−1	1/6
1/12	1/6	1/12

Figure 3.7: The multiplication weights used to calculate the Laplacian. The centre is the weight for the current position, and the others are for its neighbours.

the Euler-Lagrange equations are defined individually in Appendix B. This leads to

$$E_x(E_x u + E_y v + E_t) - \alpha^2 \nabla^2 u - \beta^2(u_{xx} + v_{yx} + w_{zx}) = 0, \tag{3.21}$$
$$E_y(E_x u + E_y v + E_t) - \alpha^2 \nabla^2 v - \beta^2(u_{xy} + v_{yy} + w_{zy}) = 0. \tag{3.22}$$

To solve these equations we need an estimate for the Laplacians of u and v. For a point at position (i, j) at time k we use the following approximation

$$\nabla^2 u \approx \bar{u}(x, y, t) - u(x, y, t), \tag{3.23}$$
$$\nabla^2 v \approx \bar{v}(x, y, t) - v(x, y, t), \tag{3.24}$$

where $\bar{u}$ and $\bar{v}$ are the local averages defined as

$$\begin{aligned}\bar{u}(x, y, t) = \tfrac{1}{6}(u(x-1, y, t) + u(x, y+1, t) + u(x+1, y, t) \\ +u(x, y-1, t)) + \tfrac{1}{12}(u(x-1, y-1, t) + u(x-1, y+1, t) \\ +u(x+1, y+1, t) + u(x+1, y-1, t)),\end{aligned} \tag{3.25}$$

$$\begin{aligned}\bar{v}(x, y, t) = \tfrac{1}{6}(v(x-1, y, t) + v(x, y+1, t) + v(x+1, y, t) \\ +v(x, y-1, t)) + \tfrac{1}{12}(v(x-1, y-1, t) + v(x-1, y+1, t) \\ +v(x+1, y+1, t) + v(x+1, y-1, t)).\end{aligned} \tag{3.26}$$

The assignment of weights used for computing the Laplacian is illustrated in Figure 3.7, these are the same weights used by Horn and Schunck [39].

We also require an estimate of the second derivatives u_{xx} and v_{yy}. These can similarly be approximated as

$$\frac{\partial^2 u(x,y,t)}{\partial x^2} \approx \tilde{u}(x,y,t) - u(x,y,t), \tag{3.27}$$

$$\frac{\partial^2 v(x,y,t)}{\partial y^2} \approx \tilde{v}(x,y,t) - v(x,y,t), \tag{3.28}$$

where $\tilde{u}$ and $\tilde{v}$ are defined as

$$\tilde{u}(x,y,t) = \frac{u(x-1,y,t) + u(x+1,y,t)}{2}, \tag{3.29}$$

$$\tilde{v}(x,y,t) = \frac{v(x,y-1,t) + v(x,y+1,t)}{2}. \tag{3.30}$$

The derivatives u_{xy}, v_{yx}, w_{zx} and w_{zy} are calculated as

$$\begin{aligned} u_{xy}(x,y,t) = \frac{\partial}{y}\frac{\partial u}{\partial x} =& \tfrac{1}{4}(u(x+1,y+1,t) + u(x-1,y-1,t) \\ & - u(x+1,y-1,t) - u(x-1,y+1,t)), \end{aligned} \tag{3.31}$$

$$\begin{aligned} v_{yx}(x,y,t) = \frac{\partial}{x}\frac{\partial v}{\partial y} =& \tfrac{1}{4}(v(x+1,y+1,t) + v(x-1,y-1,t) \\ & - v(x+1,y-1,t) - v(x-1,y+1,t)), \end{aligned} \tag{3.32}$$

$$\begin{aligned} w_{zx}(x,y,t) = \frac{\partial}{x}\frac{\partial w}{\partial z} =& \tfrac{1}{4}(w(x+1,y,t+1) + w(x-1,y,t-1) \\ & - w(x+1,y,t-1) - w(x-1,y,t+1)), \end{aligned} \tag{3.33}$$

$$\begin{aligned} w_{zy}(x,y,t) = \frac{\partial}{y}\frac{\partial w}{\partial z} =& \tfrac{1}{4}(w(x,y+1,t+1) + w(x,y-1,t-1) \\ & - w(x,y+1,t-1) - w(x,y-1,t+1)). \end{aligned} \tag{3.34}$$

At the boundary of the images some of the required neighbours will lie outside the image. In this situation we simply copy the required values from the nearest boundary position.

Substituting the definitions for the Laplacians and second derivatives into Equations 3.21 and 3.22 gives the following pair of linear equations

$$(E_x^2 + \alpha^2 + \beta^2)u + E_x E_y v = \alpha^2 \bar{u} + \beta^2(\tilde{u} + v_{yx} + w_{zx}) - E_x E_t, \tag{3.35}$$

$$E_x E_y u + (E_y^2 + \alpha^2 + \beta^2)v = \alpha^2 \bar{v} + \beta^2(\tilde{v} + u_{xy} + w_{zy}) - E_y E_t, \tag{3.36}$$

which can be written in matrix form as

$$\begin{bmatrix} E_x^2+\alpha^2+\beta^2 & E_xE_y \\ E_xE_y & E_y^2+\alpha^2+\beta^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha^2\bar{u}+\beta^2(\tilde{u}+v_{yx}+w_{zx})-E_xE_t \\ \alpha^2\bar{v}+\beta^2(\tilde{v}+u_{xy}+w_{zy})-E_yE_t \end{bmatrix}. \tag{3.37}$$

These linear equations can be solved iteratively using the Gauss-Seidel method. Each iteration estimates a new set of velocities (u^{n+1}, v^{n+1}) using the previous velocity estimates (u^n, v^n)

$$\begin{aligned} u^{n+1} = (\beta^2((E_y^2+\beta^2+\alpha^2)(\tilde{u}^n+v_{yx}^n+w_{zx}^n)-E_xE_y(\tilde{v}^n+u_{xy}^n+w_{yz}^n)) \\ +\alpha^2(\bar{u}^n(E_y^2+\beta^2+\alpha^2)-E_xE_y\bar{v}^n)-(\alpha^2+\beta^2)E_xE_t) \\ /((\alpha^2+\beta^2)(E_x^2+E_y^2+\alpha^2+\beta^2)) \end{aligned} \tag{3.38}$$

$$\begin{aligned} v^{n+1} = (\beta^2((Ex^2+\beta^2+\alpha^2)(\tilde{v}^n+u_{xy}^n+w_{zy}^n)-E_xE_y(\tilde{u}^n+v_{yx}^n+w_{zx}^n)) \\ +\alpha^2(\bar{v}^n(E_x^2+\beta^2+\alpha^2)-E_xE_y\bar{u}^n)-(\alpha^2+\beta^2)E_yE_t) \\ /((\alpha^2+\beta^2)(E_x^2+E_y^2+\alpha^2+\beta^2)). \end{aligned} \tag{3.39}$$

The initial velocities (u^0, v^0) are set to zero at every pixel position. This process is terminated when the change in velocity is less than a user defined value ϵ. We set ϵ to 0.001 for all our results. This value was empirically derived to produce good results, whilst not causing a bottleneck.

As mentioned previously, to be able to produce highly plausible animations we need the surface geometry to be closely consistent with the velocity. Once we have calculated the velocity field we use the shape-from-shading result as frame one of the height field, and then recalculate each subsequent frame in turn as follows

$$h(t+1) = h(t) - h(t)(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}). \tag{3.40}$$

This ensures the coherence between height and velocity.

3.4 Extensions

We have also created an extension to our velocity tracking approach. This extension was led by a colleague in parallel with other research we present in

this thesis, which is why we now explain its details separately to the original approach we have already described.

This extension consists of three changes to our original method. The first change made is that the advection component has not been removed from Equation 3.1. This changes the definition of $\partial w/\partial z$ used in the incompressibility condition (Equation 3.5) to

$$\frac{\partial w}{\partial z} = (\frac{\partial h}{\partial t} + \frac{\partial h}{\partial x}u + \frac{\partial h}{\partial y}v)/h. \tag{3.41}$$

Equations 3.17 and 3.40 are subsequently updated, using central differences to calculate the gradients of h.

Secondly this extension removes the image brightness condition, producing the following updated error to be minimised

$$\xi^2 = \int\int(\alpha^2\xi_c^2 + \beta^2\xi_d^2)\mathrm{d}x\mathrm{d}y. \tag{3.42}$$

This result is that the video frames are only directly used when reconstructing the geometry using shape-from-shading. The reconstructed height field is then the only information used by the optical flow algorithm. The focus is therefore solely on producing a plausible velocity field consistent with the surface geometry, rather than estimating the movement of image features between video frames.

The final extension is that multi-layer optimisation is used to obtain a velocity field which matches the height field obtained by shape-from-shading more closely. This is accomplished by recovering the velocity from both the shape-from-shading surface and several residual surfaces of different scales. These surfaces form a pyramid of target surfaces $(h^0_{\mathrm{Tar}}(t), h^1_{\mathrm{Tar}}(t), ..., h^N_{\mathrm{Tar}}(t))$ for each frame t. The target surface for the 0^{th} layer for all frames is equal to the shape-from-shading result. To generate the next layer, first the velocities for the 0^{th} layer (u^0, v^0) are calculated from the target surfaces. A final reconstructed surface $h^0_{\mathrm{Rec}}(t)$ for each frame of this layer is then computed from these velocities. A residual surface for each frame is then computed as $h^0_{\mathrm{Tar}}(t) - h^0_{\mathrm{Rec}}(t)$. The residual surface represents a low frequency velocity field that has not been captured on the current layer. To obtain this velocity field, the residual surfaces are then resized by half and used as the target surfaces $h^1_{\mathrm{Tar}}(t)$ for the next layer. The process is then repeated until the residual surfaces begin to increase, indicating an over-fitting. The final

reconstructed surface and velocities for each frame are then calculated by summing together the reconstructions from each layer of the pyramid

$$h(t) = \sum_{i=0}^{N} h^i_{\text{Rec}}(t), \tag{3.43}$$

$$u(t) = \sum_{i=0}^{N} u^i(t), \tag{3.44}$$

$$v(t) = \sum_{i=0}^{N} v^i(t). \tag{3.45}$$

This extension offers a velocity field which conforms to a more sophisticated physical constraint, to try and produce a velocity field which has greater physical plausibility. This also allows for the removal of the brightness constraint, which allows the method to calculate a velocity field which more closely matches the shape-from-shading surface. Using a multi-layer approach to minimise the error also further increases the level of fit to the shape-from-shading surface. This is because each of the different residual layers we minimise against represents a low frequency velocity field not captured by the previous layer, and therefore using the multi-layer approach allows us to capture frequencies which are lost in our original method. We will show how these extensions compare to our original method in Section 3.6.

3.5 Alternative Method of Capture

We have used videos to reconstruct water surfaces as the overall aim of our work is to allow a user to design novel scenes using videos as visual parameters, however, using the extension described in Section 3.4, our approach to calculating a velocity field will work with a height field captured from any source. Work has been carried out by colleagues [49] to capture a height field reconstruction of a water surface using a commercial dynamic 3D capture system from 3dMD [1] in a laboratory setting. The capturing system reconstructs a surface by projecting an infra-red speckle pattern onto the object of interest. This is viewed by a pair of infra-red cameras, and the speckle pattern is used internally by the system to estimate correspondences for stereo reconstruction. The system is used to capture the surface geometry of water inside a plastic container with an area of 80cm by 55cm. To allow

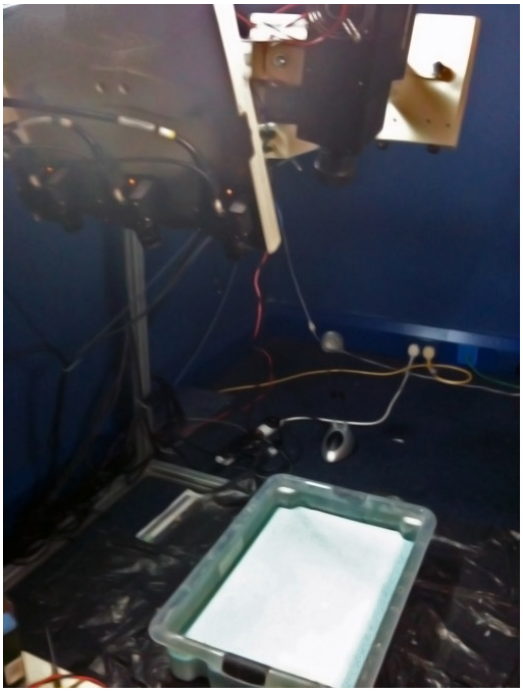


Figure 3.8: Surface scanning setup.

the speckle pattern to be viewed more reliably, white paint was added to the water similarly to Wang et al. [94].

Two sequences were captured. In the first, air was blown onto the water surface using a hair dryer to produce movement of the surface. In the second, movement was produced by moving a plastic spatula back and forth, by hand, along the edge of the water tank. We show both the captured surface, and the velocity field and recomputed height field produced by our approach in Figure 3.9. The colour code we use for visualising the velocity fields is given in Figure 3.10. These results demonstrate that we are able to calculate a velocity field which closely matches the scanned surface data. This shows that our method of obtaining a velocity field is applicable to more than just reconstruction from video, however video fits the aims of this thesis and therefore that is what we use in the rest of our work.

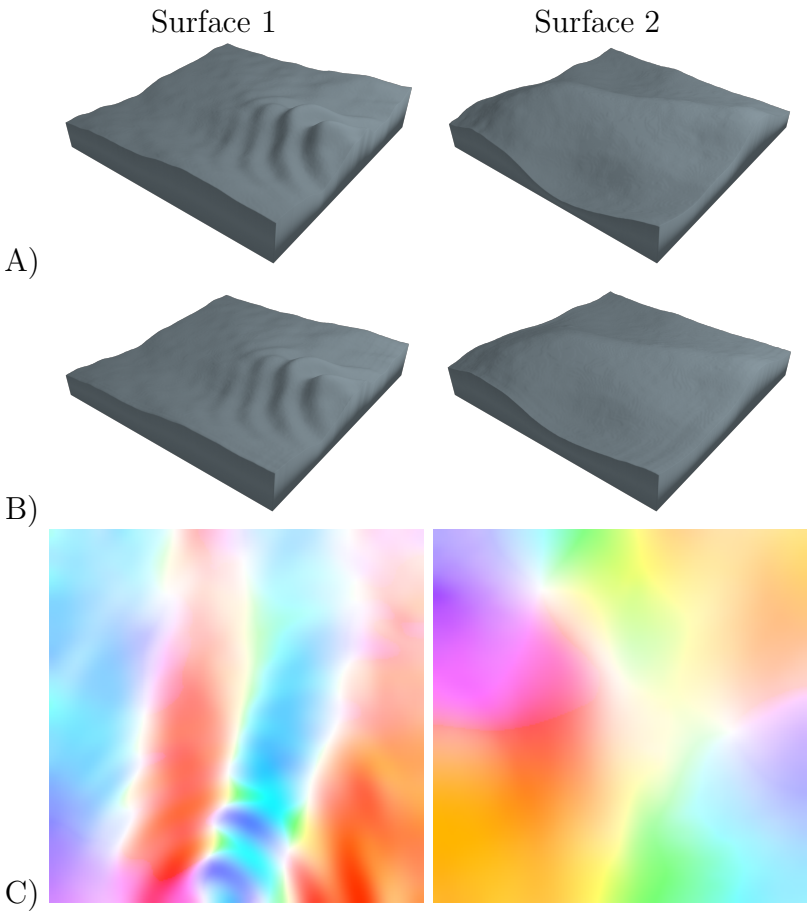


Figure 3.9: Results produced by Infra Red scanning for two different water surfaces. A) Surface produced by Infra Red scan. B) Surface reconstructed from velocity field. C) Velocity field.

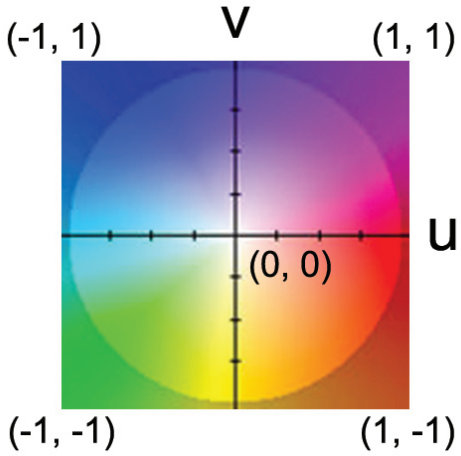


Figure 3.10: Colour code used to visualise velocity fields. This colour coding scheme was given by Baker et al. [5].

3.6 Results

Here we will present and evaluate the results of our algorithm. First we will qualitatively and quantitatively compare the effectiveness of our basic method against the extensions outlined in Section 3.1 at producing a velocity field that is consistent with the shape-from-shading surface. Secondly we will qualitatively compare both our basic method and extensions against the original Horn and Schunck [39] optical flow algorithm and a previous method designed to track water surfaces [63]. Finally we will show our reconstruction results for several example videos.

We have already shown some examples of the results produced by shape-from-shading (Figures 3.4 and 3.5). Our aim when using the optical flow tracker is to produce a plausible velocity field, which is consistent with the shape-from-shading height field reconstruction. The basic method we have described does this by using a physical constraint based on the incompressibility condition. Two extensions have also been described. The first changes the optical flow error function by removing the image brightness constraint, and extending the physical constraint to include advection. The second extension uses multi-layer optimisation to produce a better match to the shape-from-shading data. Here we evaluate how each of these tracking methods performs at calculating a velocity field which is consistent with the shape-from-shading height field. We do this by comparing each of the height

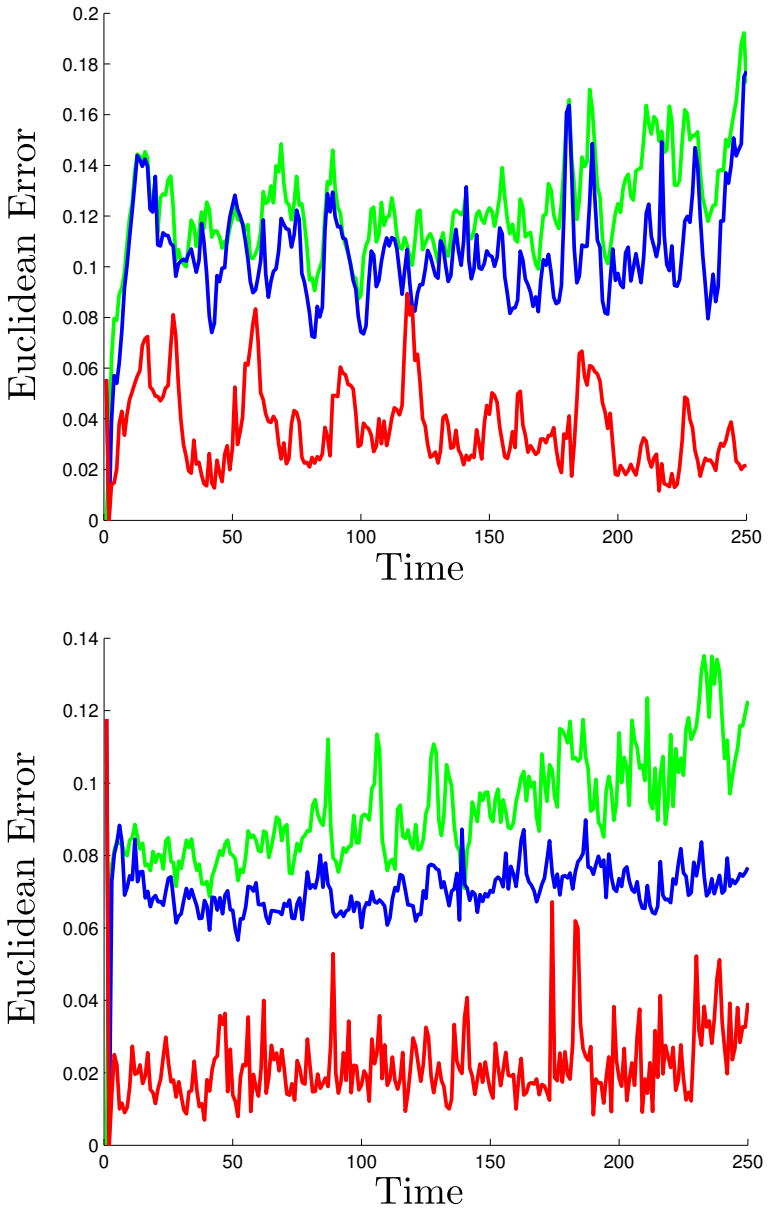


Figure 3.11: Quantitative comparison of height field results against shape-from-shading. Green) Incompressibility. Blue) Incompressibility + advection. Red) Multi-layer optimisation.

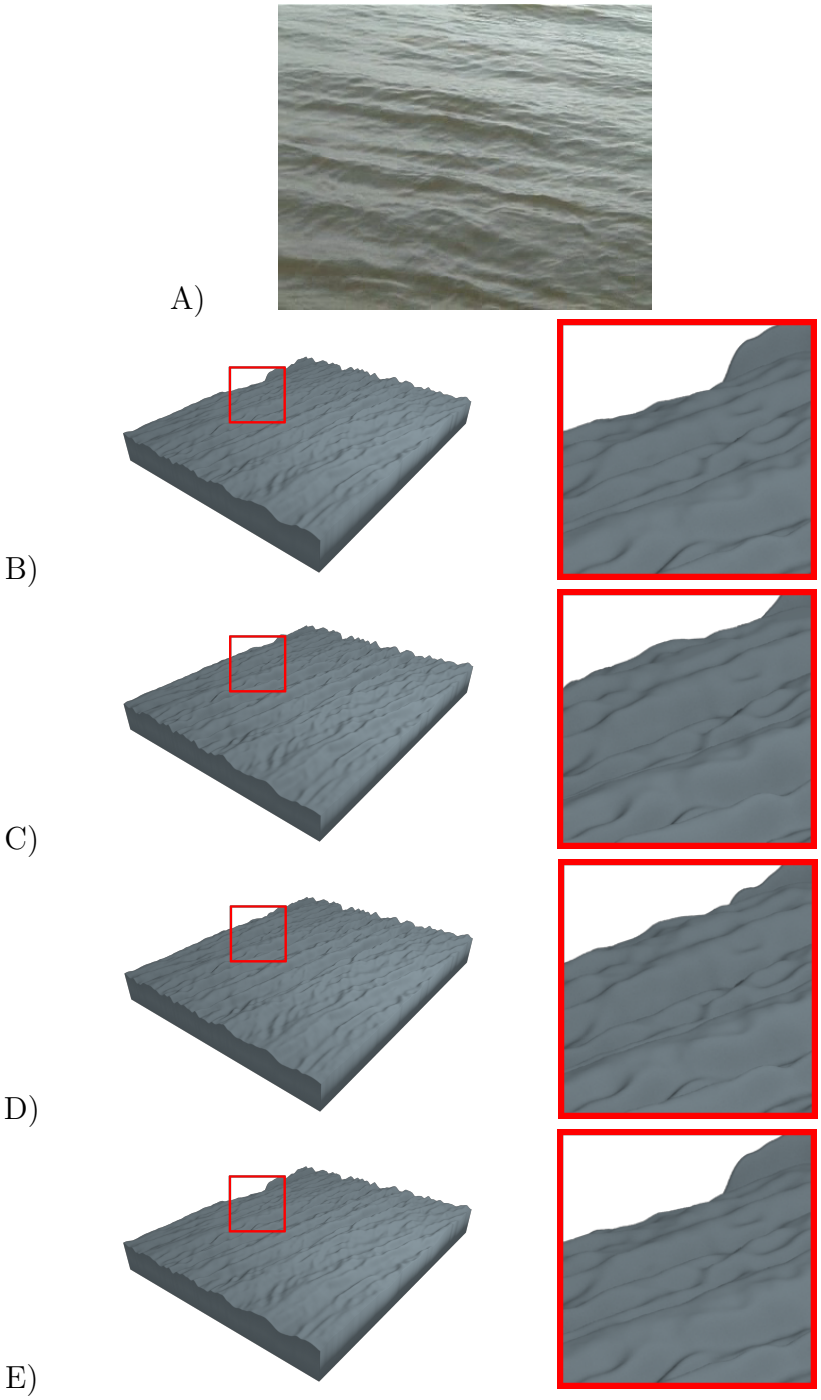


Figure 3.12: Visual comparison of height field results. A) Input. B) Shape-from-shading. C) Incompressibility. D) Incompressibility + advection. E) Multi-layer optimisation. The area within the red box is shown magnified for clarity.

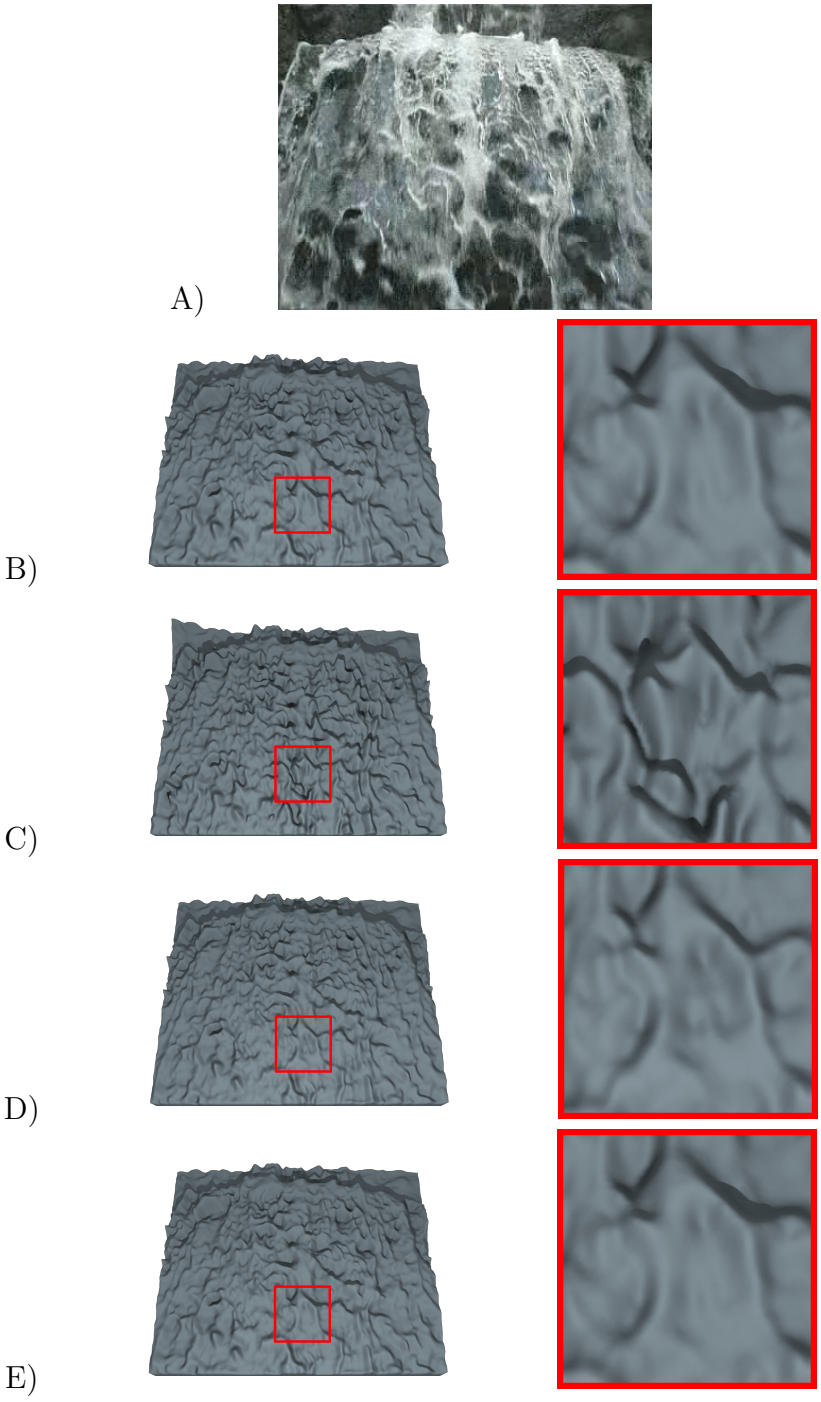


Figure 3.13: Visual comparison of height field results. A) Input. B) Shape-from-shading. C) Incompressibility. D) Incompressibility + advection. E) Multi-layer optimisation. The area within the red box is shown magnified for clarity.

fields calculated from each of the velocity field results, using the associated physical link, with the shape-from-shading surface height field. We do this comparison for each frame individually. The measure we use for comparison is the average Euclidean distance between the height values of the two frames, calculated as

$$\mathrm{ED}(h_{\mathrm{SFS}}, h_{\mathrm{Model}}) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} |h_{\mathrm{SFS}}(i,j) - h_{\mathrm{Model}}(i,j)|, \tag{3.46}$$

where M and N are the number of grid points in x and y respectively, and h_{SFS} and h_{Model} are the shape-from-shading surface and the surface calculated from the velocity field. Figure 3.11 shows the results of these comparisons for two input videos. This figure demonstrates that the worst fit to the shape-from-shading surface is calculated by our basic method based on incompressibility alone. Introducing advection and removing the image brightness constraint results in a small improvement for the first video, and a more significant improvement for the second. Using multi-layer optimisation results in a more significant improvement for both videos.

We also qualitatively compare the height field produced by the different velocity fields to evaluate whether there is any visual difference in the results. Figures 3.12 and 3.13 show the last frame of the shape-from-shading surface and each of the surfaces computed from the different velocity fields for the same two videos used in the quantitative comparison. The figure shows a clear visual difference in the results, especially in the second video (Figure 3.13). As we would expect from the quantitative comparison, the largest improvement over our basic method is produced by the multi-layer approach.

The second video used in these two comparisons displays a waterfall and is therefore an example where the shallow water assumption made by our method is clearly violated. Both the height field representation of the surface geometry of the water in this scene, and the velocity field will be physically inaccurate. However the surface geometry provides a visually plausible representation of the waterfall, and we are able to produce a velocity field which is coherent with the height field. We use this result in many of our novel scenes (Chapter 4), producing very pleasing results. The velocity field however (Figure 3.14) does not represent the perceived movement as accurately, as it does not contain the visible global downward movement of the falling water. We therefore allow the user to add a global velocity to the velocity field to manually inject this effect, whilst keeping the interesting high frequency details of our original result. This is discussed further in Chapter 4.

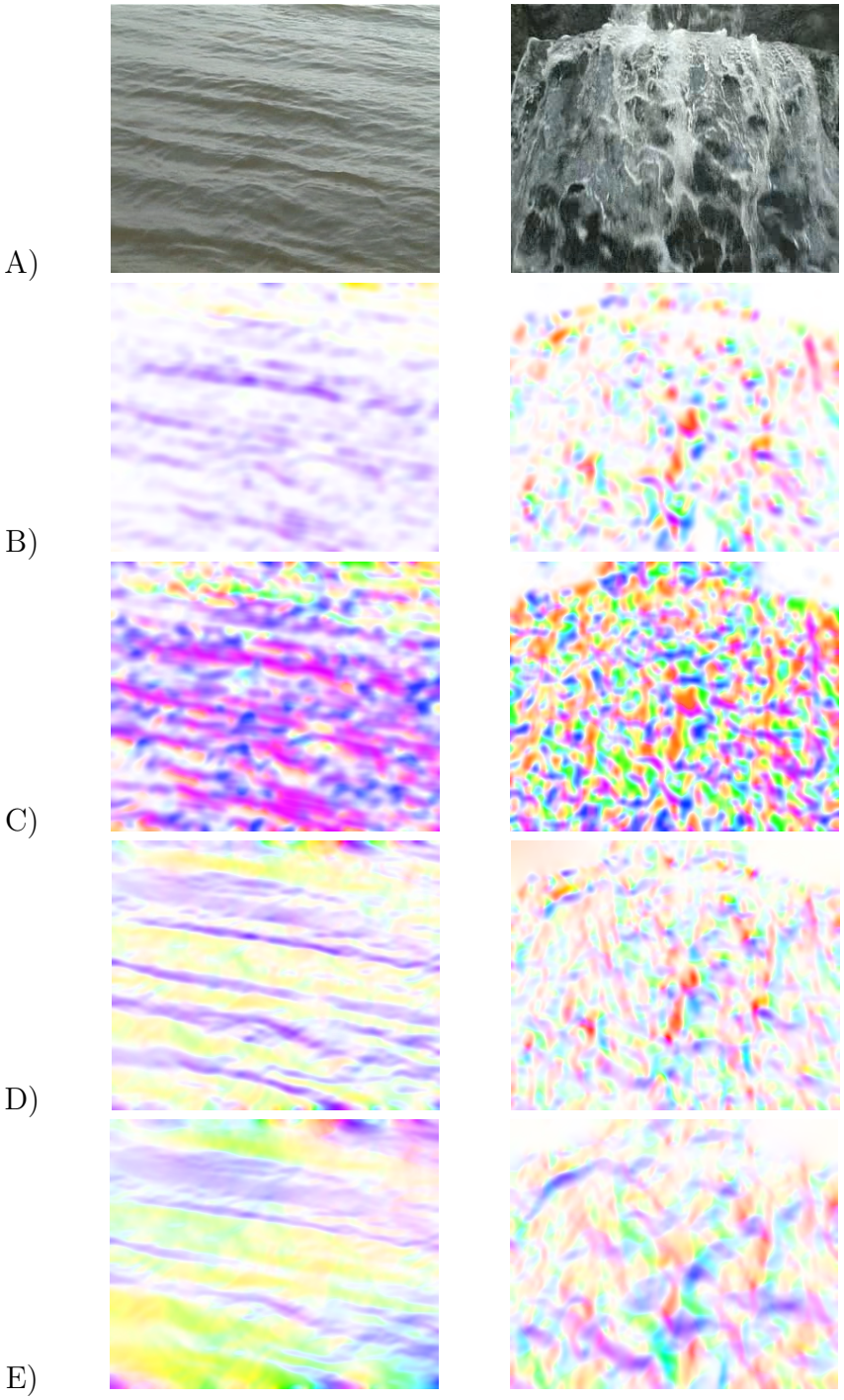


Figure 3.14: Visual comparison of velocity field results. A) Input. B) Horn and Schunck. C) Nakajima et al. D) Incompressibility. E) Incompressibility+advection with multi-layer optimisation. The shape of the waves are more visible in our results (D and E), as our velocity fields conform to the vertical surface movement.

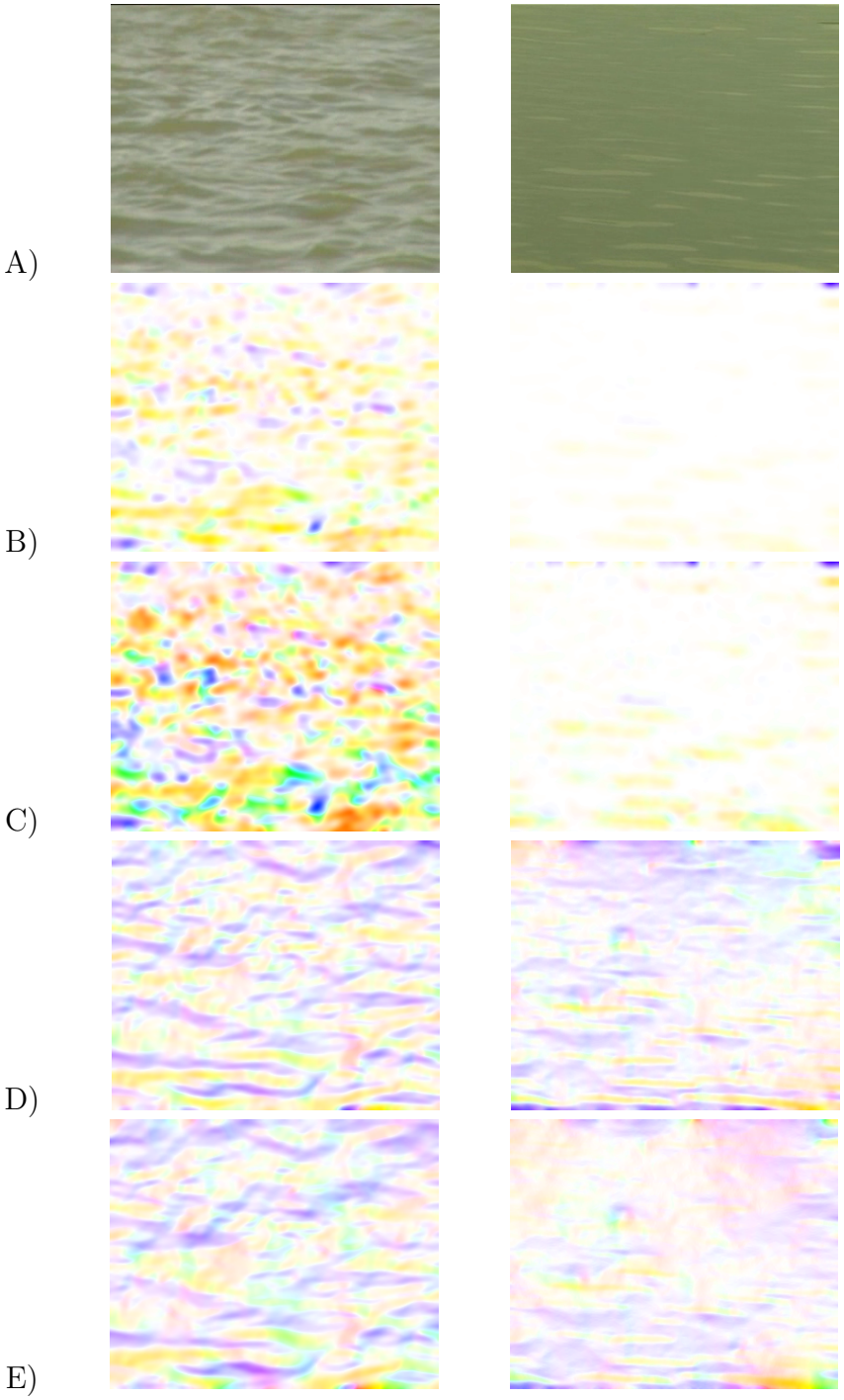


Figure 3.15: Visual comparison of velocity field results. A) Input. B) Horn and Schunck. C) Nakajima et al. D) Incompressibility. E) Incompressibility+advection with multi-layer optimisation. The shape of the waves are more visible in our results (D and E), as our velocity fields conform to the vertical surface movement.

Figures 3.14 and 3.15 show a visual comparison between the velocity fields produced by our basic incompressibility based method and our fully extended multi-layer approach for four different videos. We also show the velocity tracking results of Horn and Schunck's original approach [39], and another method designed to track water by Nakajima et al. [63]. We can not quantitatively assess the accuracy of any of the velocity fields, as we do not have ground truth data and our main aim is to produce a physically plausible velocity field that closely matches the perceived water surface geometry. The results clearly show that the standard optical flow technique by Horn and Schunck produces an over-smoothed velocity field. This is because water contains few distinct visual features which can be easily tracked by the brightness constraint. Nakajima et al. [63] extend this basic approach by adding an additional constraint based on the incompressibility condition and momentum equation from the Navier-Stokes equations. To achieve this they limit the Navier-Stokes equations to two-dimensions. Their velocity results fail to closely match the perceived water surface geometry, and can also suffer from an over-smoothed result which is clearly shown in the second video of Figure 3.15. Our approaches both produce a velocity field which appears to closely match the perceived water surface geometry. The shape of the waves can be clearly seen in our velocity images, as the velocities either side of the wave peaks are moving towards or away from the peak of the wave, depending on whether the wave is rising or falling. This is because we use the shape-from-shading result as input to our method, and use the vertical movement to constrain the optical flow result. Our method also manages to avoid over-smoothing the velocities. Our two approaches produce very similar velocity fields, but our fully extended multi-layer method produces a slightly smoother flow than our basic incompressibility method. As we have already demonstrated in Figures 3.11, 3.12 and 3.13 our fully extended method produces a closer match to the shape-from-shading surface.

Figures 3.17 and 3.18 present a wide selection of final results produced by our multi-layer approach with both incompressibility and advection included within the function describing the relationship between height and velocity. Both the velocity and height field surfaces are shown, we also show the height fields textured with the original video for increased realism. These results demonstrate our ability to reconstruct a wide range of water surfaces, some of which exhibit complex behaviour (e.g. boiling water, waterfall) which are difficult to animate with existing methods (Chapter 2).

We demonstrate some examples where our method fails in Figure 3.19. The first example shown has already been discussed in Section 3.2, and fails due to

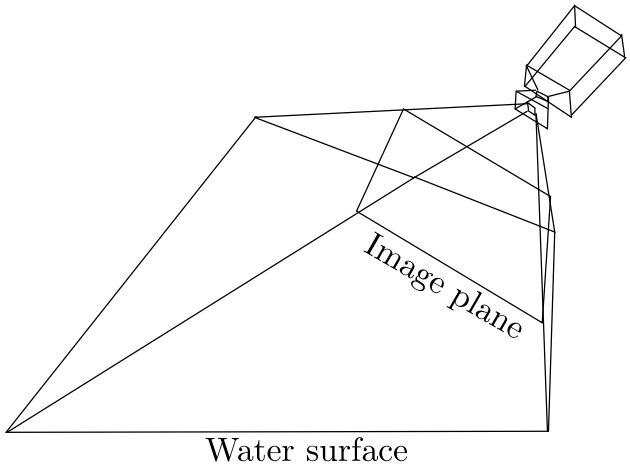


Figure 3.16: In many examples the camera direction is not orthogonal to the water surface.

the strong reflection of a tree on the water's surface. We are able to produce a plausible reconstruction for the left half of the video for the second example shown. The waves in the right half of the video are less visible due to strong shadows cast by a large structure, which results in a flat surface. The final example fails due to the incorrect angle of the camera, strong shadowing, and the poor quality of the video footage.

As we reconstruct a rectangular water surface by calculating a height for every pixel of a video, we make the assumption that the direction the camera is pointing is orthogonal to the water surface. In many of our examples the camera is not at this perfect position, but views the surface at an angle (Figure 3.16). In some cases this can cause artefacts due to the camera's perspective view of the surface. We leave correcting this to future work, as we are still able to use these results to produce high quality novel scenes (Chapter 4).

Most of the videos we reconstruct have a resolution of 352x288 pixels. Our C++ implementation of our basic method is able to compute a velocity field for approximately ten frames each second. Our fully extended approach is approximately twice as slow, mostly due to the multi-layer approach.

Videos for a selection of our final results are included on the accompanying DVD.

3.7 Conclusion

We have developed a method of reconstructing both a time-varying height and velocity field from video footage of a water surface in a natural outdoor scene. Our method first uses shape-from-shading to reconstruct the water surface, and then uses a function describing the relationship between the height and horizontal velocities to produce a velocity field with a constrained optical flow algorithm. Water scenes violate the shape-from-shading assumptions, but although our results may not be physically accurate they fulfil our aim of visual plausibility. The physical model we use also does not accurately represent all our input videos, but it successfully produces a plausible velocity field which is consistent with the height field. In some cases the velocity field fails to capture the global flow of the water, but still produces a useful representation of the local fine detailed movement. The camera's perspective also affects our results, but not to a degree that we have required a solution to this problem to use the resulting surfaces effectively when authoring novel scenes using the method reported in Chapter 4. Although previous methods sometimes produce physically accurate reconstructions, they also require a complicated experimental setup and therefore do not work on natural scenes (Chapter 2). The main advantage of our work is that a complex experimental setup is not needed, which makes it capable of reconstructing natural scenes. Our method fulfils our requirements well, whilst leaving room for interesting future work.

The next step of our work is to produce a method of using these water surface reconstructions to produce novel animations, sometimes of scenes which can not be characterised by any one of our individual reconstructions but by a combination defined by the user. This work is detailed in Chapter 4.

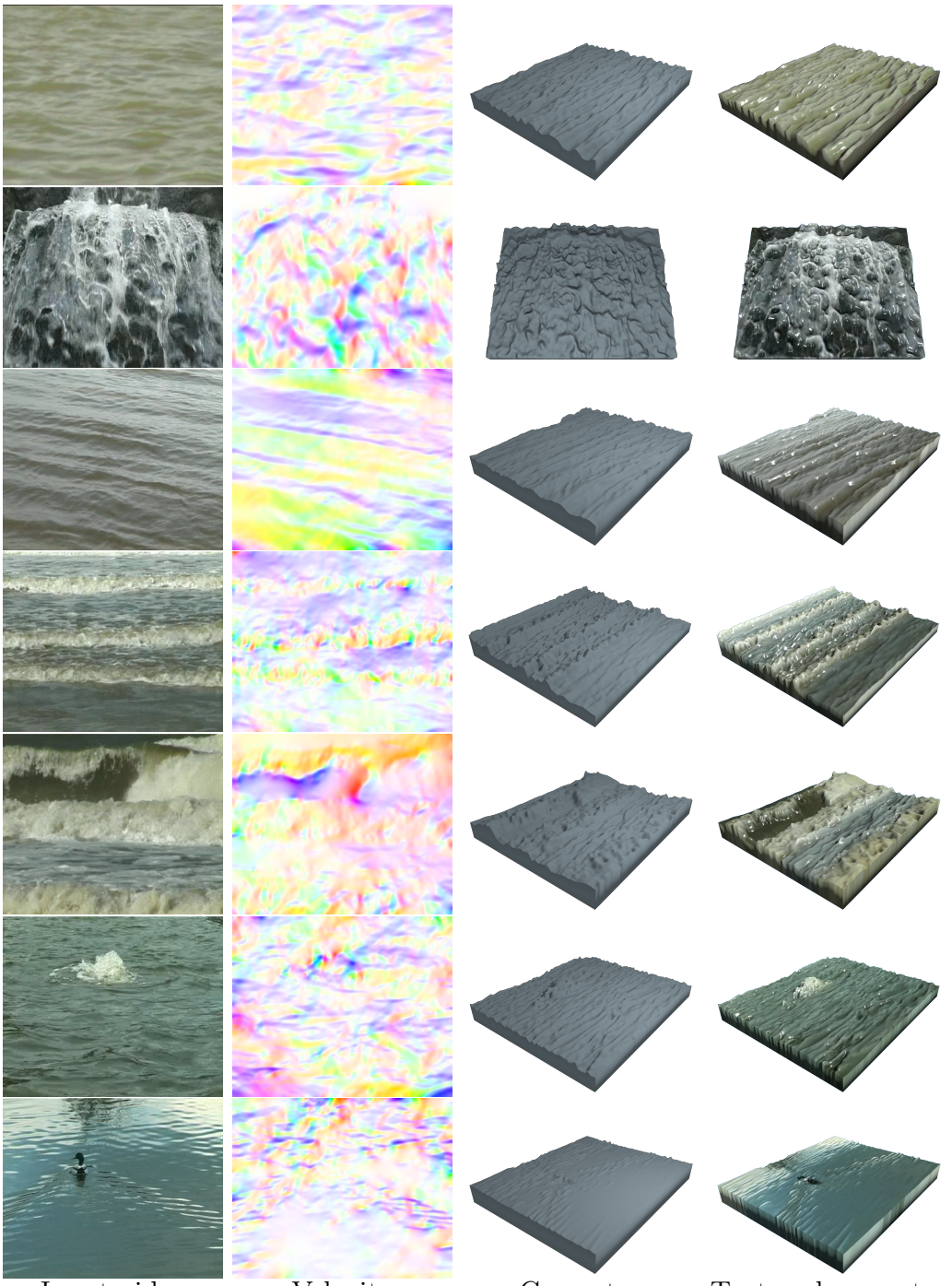


Figure 3.17: A selection of reconstruction results.

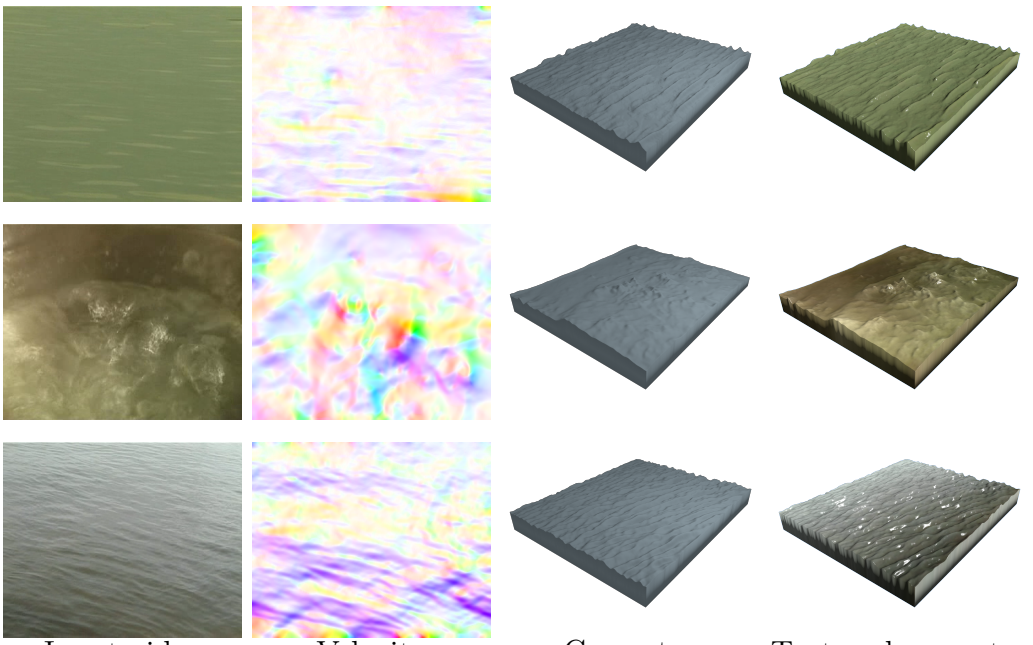


Figure 3.18: A selection of reconstruction results.

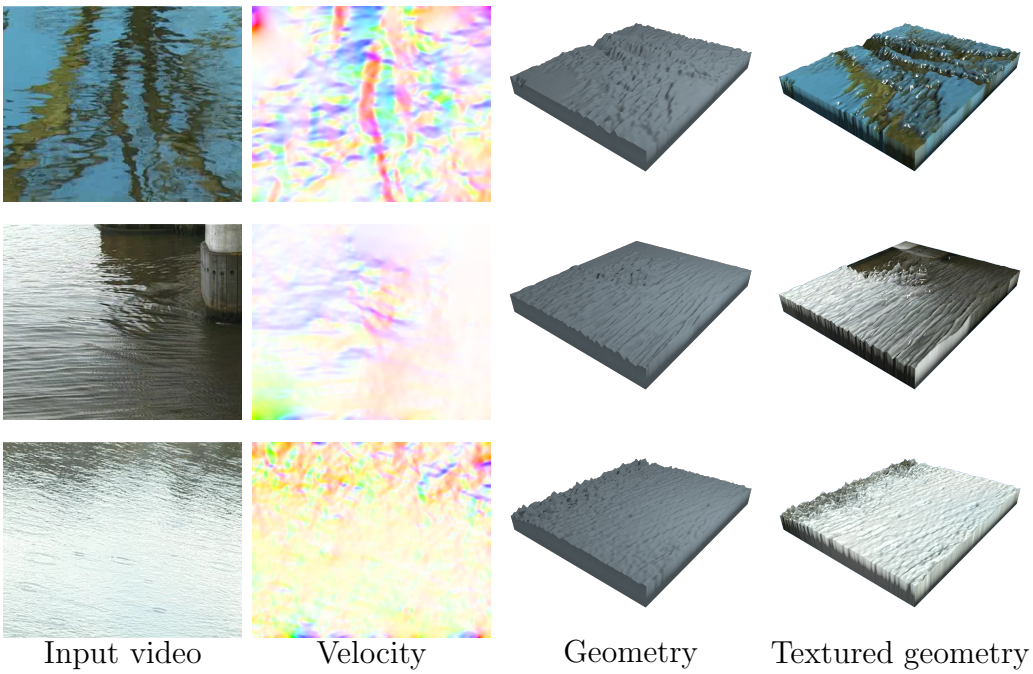


Figure 3.19: A selection of failure cases.

Chapter 4

Authoring Novel Animations of Water

The aim of our work is to produce novel animations of water from a selection of video examples. We have already described how we are able to produce reconstructions of water surfaces from video footage of natural water scenes in Chapter 3. These reconstructions consist of time-varying height and velocity fields. The aim of the work presented in this chapter is to use these reconstructed water surfaces to produce novel water animations, which are not characterised by any of the individual reconstructed surfaces but can be created by combining several surfaces together within a novel scene. In order to be able to produce a wide range of novel scenes we must be able to extend the size of a single water surface, combine multiple surfaces together seamlessly, and produce animations which loop seamlessly. We have achieved these three requirements by developing a method of using the reconstructed water patches as building blocks which can be tiled together, where each pair of adjacent patches are overlapped and blended together across this overlap region. Patches can also be composited within other patches, requiring a two dimensional blending. Real-world scenes often contain water surfaces which do not consist of straight-edged rectangles, but our height and velocity field representation easily allows us to shape our animations to emulate such scenes. To enhance the plausibility of our results, we have also developed a method of adding the effect of foam to our animations. Finally, the velocity fields can be used to drive the movement of other objects within a scene. As we combine geometry reconstructed from video footage, instead of the videos directly, the resulting water scenes are fully three-dimensional

Figure 4.1: Two surfaces can be tiled together. Top: First the patches are overlapped. Bottom: The patches are blended along a linear gradient.

and can be rendered from any viewpoint, in any rendering style.

4.1 Tiling Water Patches

To use the reconstructed water surface patches as tileable building blocks, we must be able to tile together the height and velocity fields and produce a blending from one water surface to another over space. The key to this is creating a smooth transition from one water patch to another. As discussed in Section 2.4, this is superficially similar to the work that has been done in texture synthesis. These methods can either expand the size of a single dynamic texture, or combine multiple static textures, but are unable to combine multiple dynamic textures. The reason for this is that when combining different textures these methods rely on the static structure within the image, and produce a warp from one image's structure to another. As our water surfaces consist of height and velocity fields which are animated over time, there exists no static structure which can be warped. Texture synthesis methods can therefore not be applied to our problem, and instead we have had to create our own approach which allows for the existence of the time dimension. Texture synthesis methods are also designed to work on any image textures, whereas we are able to develop a method which is tailored to the specific case of water surfaces.

Our approach to achieving a seamless blend between two water surface patches begins by placing them side by side, overlapping by a user specified amount (Figure 4.1). A blending from one water patch to another must then be

computed across the region in which they overlap. A simple approach to accomplish this blend is to use linear interpolation. If the patches are overlapped in the x direction and $x = 0$ is defined as the left border of this region we can use linear interpolation to calculate a height for every position within the overlap region as follows

$$h(x, y, t) = (1 - \alpha)h_A(x, y, t) + \alpha h_B(x, y, t), \tag{4.1}$$

where $h(x, y, t)$, $h_A(x, y, t)$ and $h_B(x, y, t)$ are the heights in the combined, left and right patches at position (x, y, t) within the overlap region. The parameter α is defined as

$$\alpha = x/N, \tag{4.2}$$

where N is the width of the overlap region.

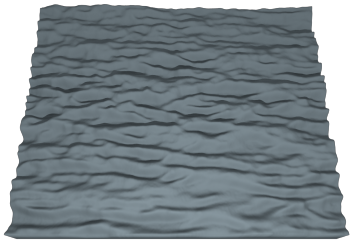
We would not expect linear interpolation to work alone, as the averaging of the two height fields would result in the damping of the height values in the overlap region. This damping would increase towards the centre of the overlap region, as the averaging increases. Our experiments confirm this, Figure 4.2 shows the result of tiling a single water surface with itself several time to produce a larger region of water. The resulting water surface shows height damping, and illustrates why we need to improve on this basic approach.

Figure 4.3 shows the result for every frame of a single position at the centre of the overlap region ($\alpha = 0.5$) of two of the water patches used in Figure 4.2. This figure demonstrates that the averaged time signal exhibits the correct shape that we want, but dampened to a smaller scale. We can therefore correct for this damping by multiplying the averaged signal by an appropriate scaling factor. This scale factor can be derived from the temporal statistics at each location of the overlap region. This is done by calculating a target standard deviation for the averaged time signal from the standard deviations of the two signals being averaged. We can then scale the averaged signal so it matches its target standard deviation.

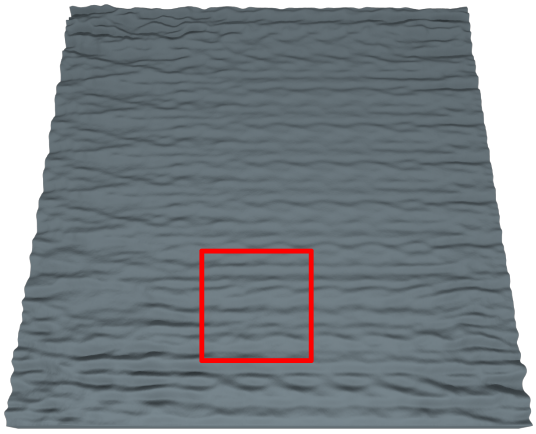
The target standard deviation $\sigma_{\text{Target}}(x, y)$ over time for every position (x, y) in the overlap region is calculated as

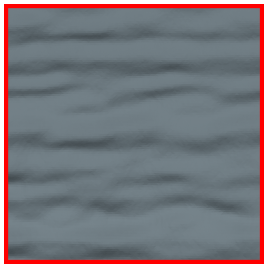
$$\sigma_{\text{Target}}(x, y) = (1 - \alpha)\sigma_A(x, y) + \alpha\sigma_B(x, y), \tag{4.3}$$

where $\sigma_A(x, y)$ and $\sigma_B(x, y)$ are the standard deviations over time at position (x, y) in the overlap region for the left and right patches respectively. The



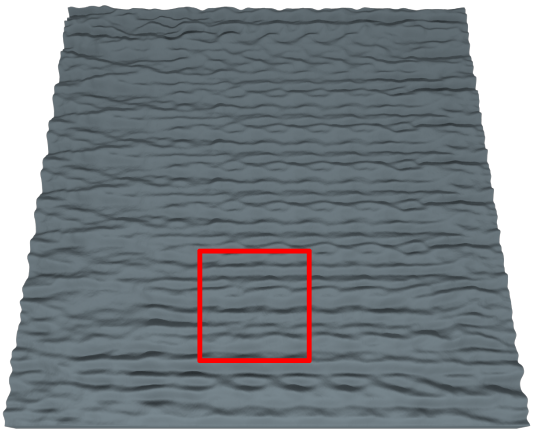
Original water surface.

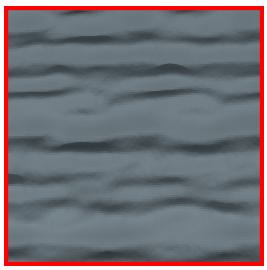




Magnification.

Extended surface shows damping.





Magnification.

Extended surface with damping corrected.

Figure 4.2: Water surface extended to a larger size. Using linear interpolation alone causes the heights to be dampened, which we correct using the surface's temporal statistics. The area within the red box is shown magnified for clarity. The spatial repetition is solved in Section 4.3

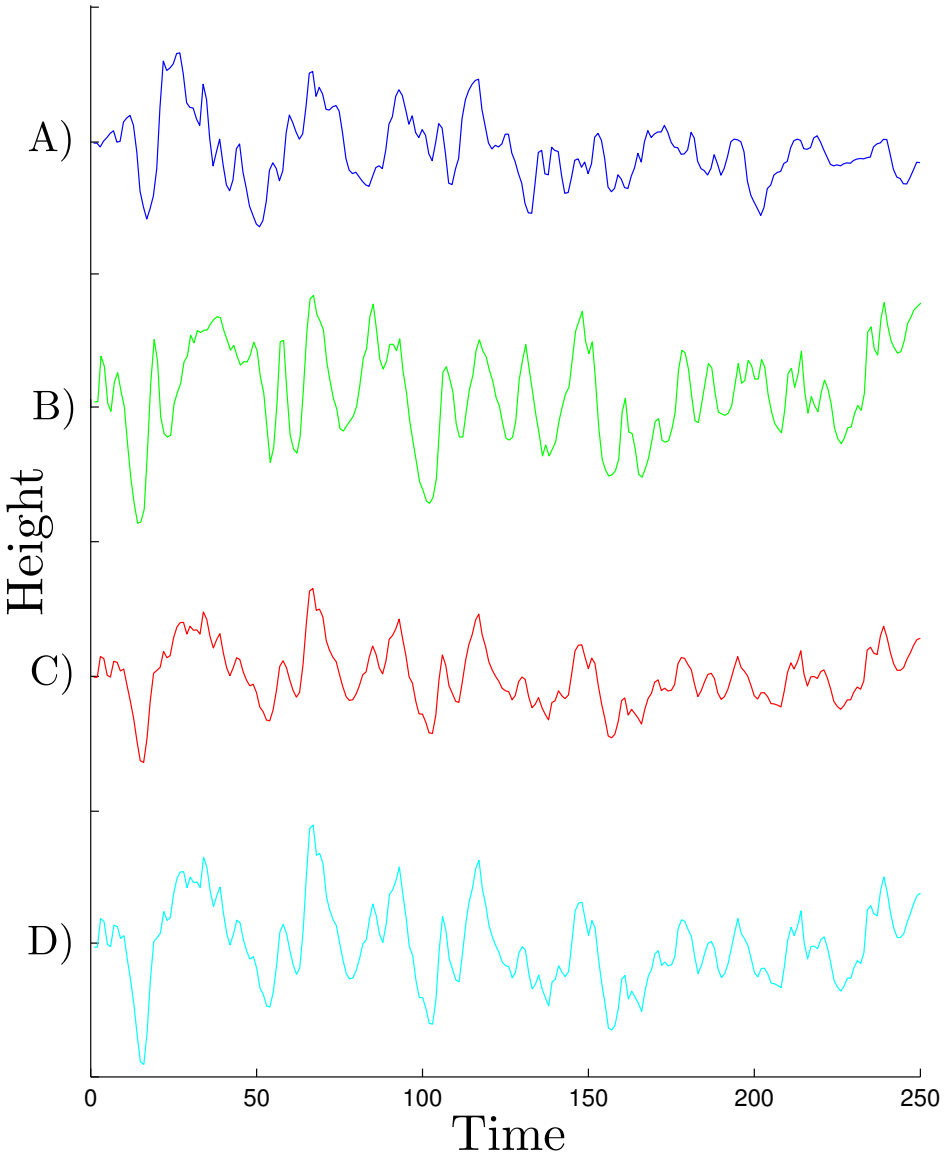


Figure 4.3: Demonstration of damping correction for a single point at the centre of the overlap region of two water patches ($\alpha = 0.5$). A) - left patch, B) - right patch, C) - combined patch using linear interpolation, D) - combined patch with damping corrected.

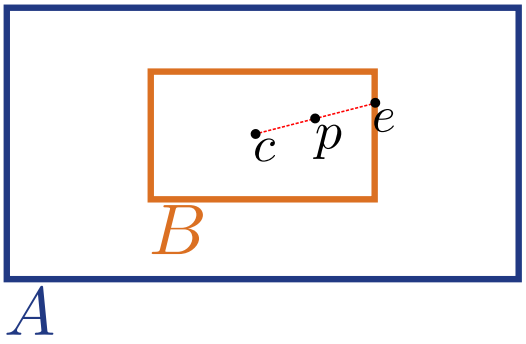


Figure 4.4: One surface B can be placed over another A. The 2D gradient used for blending equals one at the user defined centre c, and zero at the boundaries of the overlap region.

same α is used as defined in Equation 4.2. We then scale the height at each (x, y) position within the overlap region of the height field produced by Equation 4.1 so its standard deviation matches $\sigma_{\text{Target}}(x, y)$. This is done by first shifting the height values so that the mean at each of these points lies at zero

$$h'(x, y, t) = h(x, y, t) - \mu(x, y). \tag{4.4}$$

We then scale and shift back by the original mean as follows

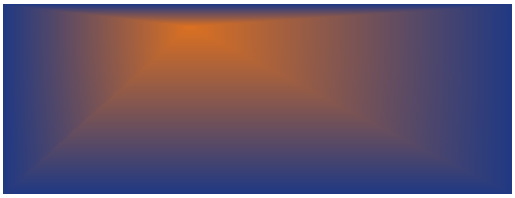
$$h''(x, y, t) = \frac{\sigma_{\text{Target}}(x, y)}{\sigma(x, y)} h'(x, y, t) + \mu(x, y), \tag{4.5}$$

where $\sigma(x, y)$ and $\mu(x, y)$ are the standard deviation and mean over time of the initial combined height field at position (x, y). Figure 4.3 demonstrates this correction for a single height field position at the centre of the overlap region ($\alpha = 0.5$) of two overlapping patches, and Figure 4.8 demonstrates the improvement of our damping correction over linear interpolation alone when tiling a water surface with itself. We describe a method for correcting the obvious spatial repetition visible in this figure in Section 4.3.

When patches are combined along the y direction they are placed vertically adjacent to one another, overlapping by the user specified amount in y. The blending between water patches is calculated using the same method by substituting x for y in Equation 4.2.

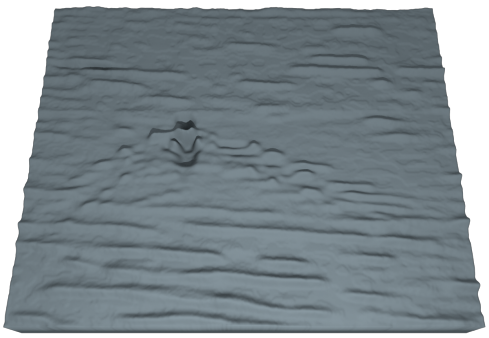
Sometimes the user may not wish to place two water surface patches side-by-side, but composite one onto another. This would require the patches to be blended in two dimensions. An example of this is presented in Figure 4.5.

Water to be composited.

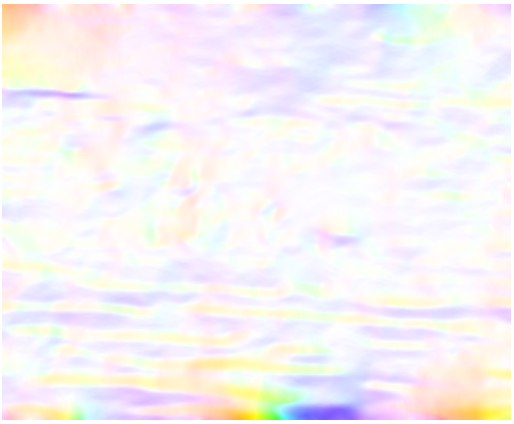


Blending gradient.

Water to be composited onto.



Resulting water surface geometry.



Resulting water surface velocity.

Figure 4.5: One water surface is composited onto another, adding the ripple effect caused by the duck onto the flowing stream.

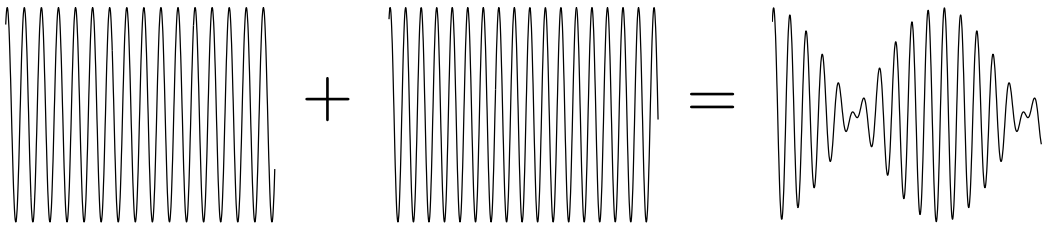


Figure 4.6: Adding two sine waves of differing frequency results in a beat wave.

To accomplish this the user places one patch B over another A (Figure 4.4). The updated heights are calculated with the same method, but changing the definition of α to

$$\alpha = \frac{|p - e|}{|c - e|}, \tag{4.6}$$

where p is the current (x, y) position in the overlap region, c is a user defined gradient centre, and e is the position on the border of the overlap region that intersects the straight line running from c through p. Although we use this square gradient for all our examples, another gradient fulfilling specific requirements could be used instead.

For the resulting animations to have a corresponding velocity field, we must also be able to tile our velocity field reconstructions. Each of the velocity field components, u and v, is blended in exactly the same way as the surface geometry by substituting it for the height values in Equations 4.1, 4.4 and 4.5.

4.2 Beat Waves

Our method of blending between two water patches calculates the weighted average of the time component of two height fields at each spatial location of an overlap region. As these two time components are essentially one-dimensional signals, they can be represented as the sum of a collection of sine waves of different frequencies. We therefore might expect that summing these two waves will exhibit the same properties as summing two different sine waves. The sum of two sine waves of equal amplitude, but differing

frequencies can be written as follows

$$\begin{aligned} y(t) =& A\sin(\omega_1 t) + A\sin(\omega_2 t) \\ =& 2A\cos(\frac{\omega_1 - \omega_2}{2}t)\sin(\frac{\omega_1 + \omega_2}{2}t), \end{aligned} \tag{4.7}$$

where t is time, A is the amplitude of each sine, and ω_1 and ω_2 are the angular frequencies of the first and second sine wave respectively. The result has two components. The first is a cosine wave which oscillates with a frequency equal to half the difference of the original frequencies, and the second is a sine wave which oscillates with the average frequency of the two original waves. The cosine term creates an amplitude "envelope", which causes the phenomenon often referred to as "beats". An example is illustrated in Figure 4.6.

We may be led to expect that a similar phenomenon may occur when we perform our blending, which would lead to undesirable artifacts. As we can see from Figure 4.3, this is not the case. Using the fast Fourier transform, two functions of time $f(t)$ and $g(t)$ can be represented as follows

$$f(t) = \sum_{u}^{N} = F(u)\mathrm{e}^{iut}, \tag{4.8}$$

$$g(t) = \sum_{u}^{N} = G(u)\mathrm{e}^{iut}. \tag{4.9}$$

Weighting and summing these can therefore be written as

$$\begin{aligned} \alpha f(t) + \beta g(t) =& \alpha \sum_{u}^{N} F(u)\mathrm{e}^{iut} + \beta \sum_{u}^{N} G(u)\mathrm{e}^{iut} \\ =& \sum_{u}^{N} (\alpha F(u) + \beta G(u))\mathrm{e}^{iut}. \end{aligned} \tag{4.10}$$

This shows that averaging these two signals is equivalent to averaging each pair of identical frequency components (Figure 4.7). Since the time signals we are averaging will always be the same length, they contain the same frequency components but with varying amplitude and phase. Therefore we are always averaging two sines with the same frequency, and hence the result does not exhibit any beating phenomena. As the phase of each pair of frequency components may not be identical, this accounts for the damping we have already discussed in Section 4.1.

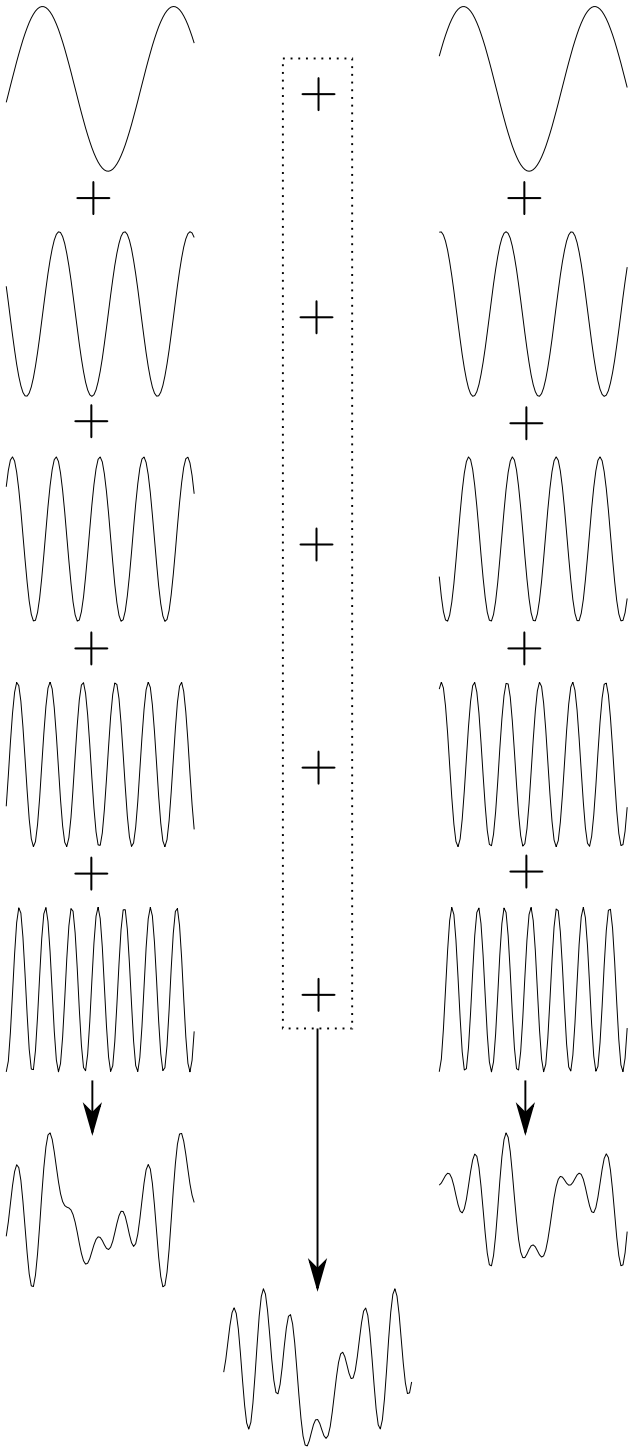


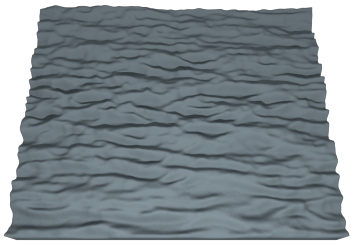
Figure 4.7: Summing two complex waves is equal to summing each of their frequency components. This results in no beat waves.

4.3 Looping and Extending

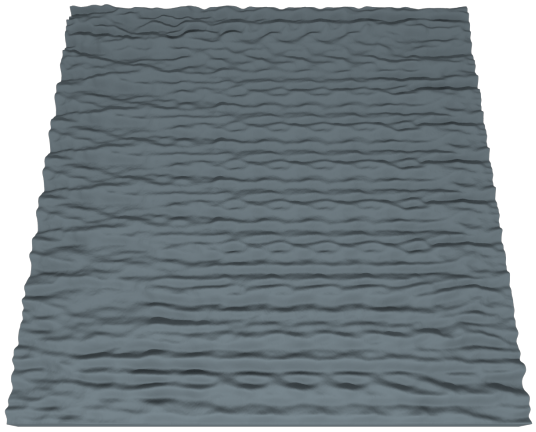
For some applications the user may require an animation which has a duration longer than the available video footage. For applications such as games, an infinitely looping animation may be needed. To solve this problem, we can create animations which can be looped seamlessly from water patches which are periodic in time. As detailed in Section 2.4, this has been done before for by Schödl et al. [79] for dynamic textures by finding sets of matching frames. They either transition between these frames during playback, or they crop the video at two matching frames which have a reasonable distance between them to produce an infinitely repeating video with a seamless loop. The drawback of this method is that sometimes the best matching frames do not match close enough, causing an undesirable result. They demonstrate their technique on scenes such as swinging pendulums, which are less likely to suffer from this problem as they contain very simple repeated motion. The motion of water is far more complex, and although it is repetitive it is less likely for two distant frames to match very closely. Therefore instead of using this existing method we take advantage of the fact that tiling can be done in time as well as space with a small adaptation to the method we presented in Section 4.1.

To tile in time, patches are first overlapped by a user defined number of frames. Secondly x is replaced with t in Equation 4.2, where $t = 0$ at the first frame of the overlap region and N is the number of frames overlapping. Lastly all (x, y) in Equations 4.3, 4.4 and 4.5 are replaced with t, and the mean and standard deviations are taken over all values in x and y. A looping animation is achieved by cutting a few frames from the beginning of the animation, overlapping them completely with the last few frames and then blending into them with this adapted method (Figure 4.9).

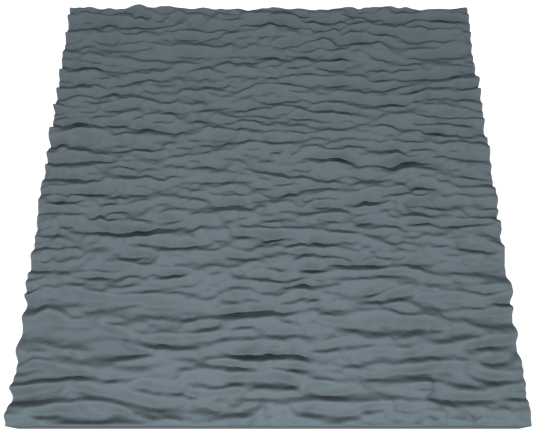
The user may also require an area of water which matches the behaviour of a single example video, but which is spatially larger than the water surface displayed within the video footage. A patch of water which displays spatial periodicity can be extended to any size by repeatedly tiling it with copies of itself. To reach the desired size, a user may have to tediously patch together many copies. We therefore automate this process. First the original patch is repeatedly tiled in the x direction, using a user defined overlap, until the desired width is reached or exceeded. This new combined patch is then tiled the same way in the y direction. If the specified overlap size between pairs of patches causes the resulting surface to be greater than the



Original water surface.



Extended patch exhibiting repetition.



Extended patch with the repetition corrected.

Figure 4.8: Naively extending a water surface produces obvious repetition, which we correct by randomising the start frame of each duplicate water patch before tiling.

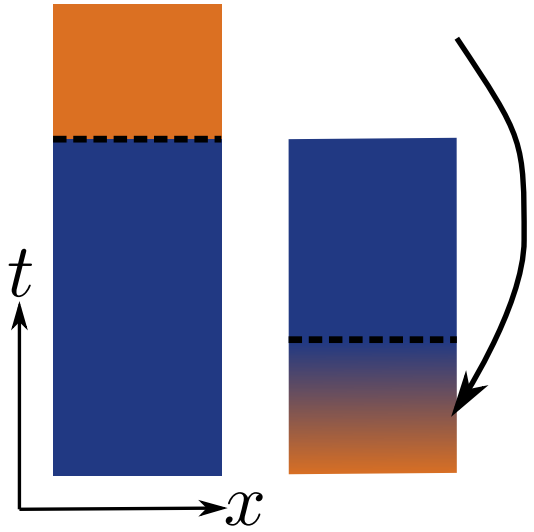


Figure 4.9: An infinitely looping animation can be created. Left: A block of frames are cut from the beginning of the video. Right: The block is placed over the end of the video and blended into it using a linear gradient.

desired size, it is simply cropped. A result of extending a water surface in this way is shown in Figure 4.8. Since this approach tiles a patch with duplicates of itself, the result contains obvious spatial repetition. We can correct for this if the original animation is temporally periodic. We alter the approach by first producing a looping animation of the original water patch. We then randomise the starting frame of each duplicate patch before blending together. This procedure eliminates spatial repetition within any individual frame. Even though the same movement may be exhibited in different surface locations at different moments in time, we choose a random start frame so that there is no visible pattern to the frame staggering. We find that this makes it is very difficult to visually detect any repetition. A single frame of an extended surface, with and without this repetition corrected, is shown in Figure 4.8.

4.4 Advecting External Objects

Many scenes contain objects floating or moving through water. As our water surface animations include velocity fields, we can use these to drive the movement of such objects. To do this an object is first placed onto the wa-

ter surface at the desired starting position. At each subsequent frame t the velocity (u, v) at the object's position (x, y) is interpolated from the four nearest grid points of the velocity field

$$\begin{aligned} u = \omega_1(\omega_3 u(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, t) + \omega_4 u(\lfloor x \rfloor + 1, \lfloor y \rfloor, t)) \\ +\omega_2(\omega_3 u(\lfloor x \rfloor, \lfloor y \rfloor + 1, t) + \omega_4 u(\lfloor x \rfloor, \lfloor y \rfloor, t)), \end{aligned} \tag{4.11}$$

$$\begin{aligned} v = \omega_1(\omega_3 v(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, t) + \omega_4 v(\lfloor x \rfloor + 1, \lfloor y \rfloor, t)) \\ +\omega_2(\omega_3 v(\lfloor x \rfloor, \lfloor y \rfloor + 1, t) + \omega_4 v(\lfloor x \rfloor, \lfloor y \rfloor, t)), \end{aligned} \tag{4.12}$$

where

$$\omega_1 = x - \lfloor x \rfloor, \tag{4.13}$$

$$\omega_2 = 1 - \omega_1, \tag{4.14}$$

$$\omega_3 = y - \lfloor y \rfloor, \tag{4.15}$$

$$\omega_4 = 1 - \omega_3. \tag{4.16}$$

$$\tag{4.17}$$

The object is moved along the surface according to the resulting velocity vector

$$x' = x + u, \tag{4.18}$$

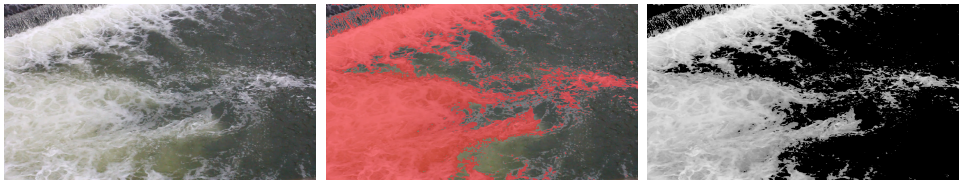
$$y' = y + v. \tag{4.19}$$

To prevent the object from leaving the water surface, if the calculated object position lies outside the velocity grid it is repositioned to the nearest boundary position.

As we explained in Chapter 3 our velocity fields do not always exhibit the global flow of the water. Also, sometimes the user may wish to add a flow direction to a velocity field, one which was not necessarily present in the source video. We therefore allow the user to add a global flow to the water, consisting of a velocity vector which is added to all the grid points of the velocity field. Global flow was added to the velocity field to produce the result shown in Figure 4.19.

4.5 Foam

Foam is an important visual characteristic of many real-world water scenes, and we therefore wish to give the user the ability to include foam in their



Video frame. Foam segmented (red). Final opacity values.

Figure 4.10: Foam is extracted from a video, using a parameter value of 0.5 to segment the video, in the form of opacity values which are applied to a white texture.

animations. As we already use videos of real water surfaces to produce the water's geometry, we can exploit this resource to reconstruct foam textures. Our existing method only produces the water surface geometry, and although foam visible within the original videos may produced raised sections on the reconstruction result, we do not deal with the foam explicitly and therefore do not replicate the effect it has on the colour appearance of the water. In the case where foam appears in the original video footage, we wish to allow the user to extract a foam texture which can be applied to the reconstructed water surface. In the case where there is no foam in the original video, we would like the user to be able to apply a foam texture to it which has been extracted from a different video.

We extract a foam texture from a video by first segmenting the foam from the rest of the video. This is done by classifying all pixels with a luminance value greater than a user defined parameter as foam. The luminance values of the pixels classified as foam are then normalised to lie in the range $[0, 1]$. The luminance values for all other pixels are set to zero. These values are used as the animated opacity values of a plain white texture. This texture is then applied to a water surface to add the appearance of foam. We tile these foam maps with the same methods used for the water surfaces. If the user only wishes to extract the foam displayed in a particular region of the video, then they can draw a mask over the first frame to indicate which portion of the video to use, and we set all pixels outside this region to black. An example is shown in Figure 4.10, where pixels with luminance values greater than 0.5 were classified as foam.

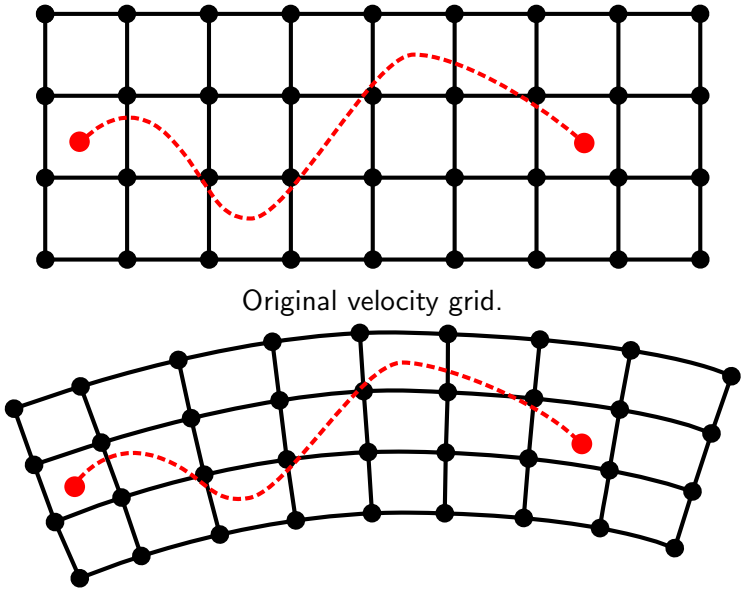


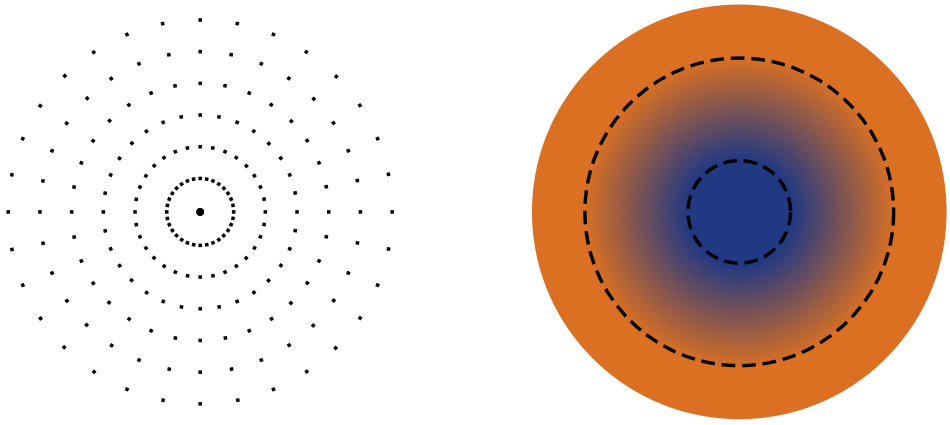
Figure 4.11: An object can be advected through the shaped velocity field, by mapping its path (shown in red) through the original grid onto the curved grid.

4.6 Shaping Surfaces

The shapes of many natural bodies of water do not consist of straight-edged rectangles. To be able to emulate these synthetically we must be able to shape the water surfaces. To achieve this, we first map a height field onto the vertices of a rectangular mesh with the same resolution as the height field. Instead of representing the heights as displacements of the surface in a specified up-axis, they are represented as a displacement in the direction of the surface normal at each vertex. This mesh can then be shaped and curved to a more desirable shape, while preserving the original water motion. To advect objects along a curved water surface, we first calculate their movement through the original rectangular velocity field. The object's position at each frame can then be mapped onto the equivalent location on the curved mesh (Figure 4.11), which is calculated as

$$\begin{aligned} p'(x,y,t) = \omega_1(\omega_3 V(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, t) + \omega_4 V(\lfloor x \rfloor + 1, \lfloor y \rfloor, t)) \\ +\omega_2(\omega_3 V(\lfloor x \rfloor, \lfloor y \rfloor + 1, t) + \omega_4 V(\lfloor x \rfloor, \lfloor y \rfloor, t)), \end{aligned} \tag{4.20}$$

where (x, y, t) is the object's position within the original velocity field, and $V(i, j, k)$ is the position of vertex (i, j) of the curved mesh at frame k. The



Circular mesh. Blending along r.

Figure 4.12: We are able to create circular water surfaces by resampling vertices in a circular layout. Rectangular patches can be wrapped around a circular mesh and the two can be blended along the radius r.

weightings ω_1, ω_2, ω_3 and ω_4 are defined as in Equations 4.13, 4.14, 4.15 and 4.16. If the water surface is curved like in Figure 4.11, this method of mapping the object's path has the effect of amplifying velocities on the outside of the bend, and decreasing them on the inside. We find that this mapping can give good results (Section 4.7), but this may not always be what the user would like. We have already discussed that changing the velocity field does not affect the surface geometry (Section 4.4), therefore the user is able to edit the velocity field to suit their specific requirements. Further work on tools for intuitive editing of the velocity field would be interesting, but we leave it for future research.

We are also able to create animations where a circular water surface is more appropriate. An example scene which uses this technique is shown in Figure 4.16. We make a circular height field by taking one of our rectangular height fields, cutting a circular area out of it and resampling the vertices in a circular layout (Figure 4.12). The reason for resampling the vertices in this way is that it allows us to easily blend with a rectangular patch that is wrapped around the outside of the circle, which is explained at the end of this section. To resample the vertices the user selects the centre point of the circle $(x_{\text{Centre}}, y_{\text{Centre}})$ on the original rectangular grid, and also defines the radius of the circle R. The number of grid points to be placed along the circle's radius is set equal to the number of grid points along the same length of the rectangular grid. The height for each new grid point (r, θ) is

calculated as

$$h_{\text{Circle}}(r,\theta) = h_{\text{Rectangle}}(r\cos\theta + x_{\text{Centre}}, r\sin\theta + y_{\text{Centre}}), \tag{4.21}$$

where $h_{\text{Circle}}(r,\theta)$ is the height at point (r,θ) of the new circular grid, and $h_{\text{Rectangle}}(x,y)$ is the height at point (x,y) of the original rectangular grid. Velocities along r and θ are also resampled at the same positions (r,θ) as

$$u_{\text{Circle}}(r,\theta) = \sqrt{x^2+y^2}, \tag{4.22}$$

$$v_{\text{Circle}}(r,\theta) = \text{atan2}(y,x), \tag{4.23}$$

where

$$(x,y) = (\vec{a}\cdot(\frac{\vec{b}}{|\vec{b}|}))\frac{\vec{b}}{|\vec{b}|}, \tag{4.24}$$

$$\vec{b} = (r\cos\theta, r\sin\theta), \tag{4.25}$$

$$\vec{a} = (u_{\text{Rectangle}}(r\cos\theta + x_{\text{Centre}}, r\sin\theta + y_{\text{Centre}}), \\ v_{\text{Rectangle}}(r\cos\theta + x_{\text{Centre}}, r\sin\theta + y_{\text{Centre}})), \tag{4.26}$$

and $u_{\text{Circle}}(r,\theta)$ and $v_{\text{Circle}}(r,\theta)$ are the velocities at point (r,θ) of the new circular grid, and $u_{\text{Rectangle}}(x,y)$ and $v_{\text{Rectangle}}(x,y)$ are the velocities at point (x,y) of the original rectangular grid.

A rectangular surface can be wrapped around the outside of the circle, overlapping by a user specified amount, and blended along the radius (Figure 4.12). This is easily accomplished by treating each ring of vertices around the circumference of the circular mesh as a row of a rectangular mesh, and blending using the method described in Section 4.1.

4.7 Results

We have used our water authoring method to make several novel scenes to demonstrate its use. The first example is displayed in Figure 4.13, and depicts a river through a hilly landscape. We used seven example videos to make this scene. Five examples were used to create the surface of the flowing river. A foam texture was also applied to this surface, but from a separate video example. Both the surface and foam texture for the waterfall were created from a single example video. Both the surface and foam texture extracted

from this video were extended to a much larger size, with the patches at the bottom of the waterfall vertically stretched before blending with the higher patches to create the effect of the water accelerating as it falls.

Our second example is a water slide (Figure 4.14), created using five example videos. The water on the slide was created from two examples and the pool beyond from three. The example video highlighted in green was used to provide a foam texture for the surface patch reconstructed from the same video.

The water feature shown in Figure 4.15 was created using just three examples. The example video highlighted in blue was used to provide a foam texture for the surface patch reconstructed from the same video. When extending the surface reconstructed from this same video example, we split the surface into two components. One component contained the "bubbly" section of the surface, the other contained the "wavy" section. The user could then specify where these two components were to be used in order to place the bubbles in the correct place beneath the waterfalls.

The scene of a lake with boiling water springs in Figure 4.17 was created using two examples. The example highlighted in green was composited onto an enlarged version of the example highlighted in blue. When compositing we added the green example to the surface, rather than blending into it. This kept the full movement of the lake, whilst including the effect of the water boiling.

To demonstrate a less linear construction, we have also used our method to make an animation of a fountain (Figure 4.16). The top upwelling pool of water feeding into the waterfall was created using two video examples, one for the centre of the circular mesh, and one which was tiled around the outer region of the circle. This mesh was then curved along the radius direction to produce the depicted shape. The water pool below was created from two water examples which were combined and then tiled around a central circle which is covered by the central pillar of the fountain. The example video highlighted in red was used to provide a foam texture, which was then applied to the same location as the reconstructed surface underneath the waterfall.

For our final two examples we show how the extracted velocity field can be used to influence other objects in the scene. The first of these is an animation of reeds swaying with the movement of water in a lake (Figure 4.18). The

second is used to demonstrate how objects can be advected along a shaped water surface (Figure 4.19). It shows several rubber ducks drifting along a flowing stream. In this example the user added a small global velocity in the direction the waves appear to be moving, so that the ducks travel along the stream.

All the animations shown have been modified to loop seamlessly in time. Most of the individual water patches we use have a resolution of 352x288 pixels, and the overlap width used when tiling is always set between 50 and 100 pixels in our examples. The quality of the blend between the water patches is not highly dependant on the overlap size, therefore this value has not had to be fine tuned to produce good results. The blending is computed in approximately 5 to 15 seconds with our Matlab implementation.

These results demonstrate that we are able to create a wide range of animations, emulating both natural and man-made scenes, producing visually plausible results.

A video for each of these water scenes is included on the accompanying DVD.

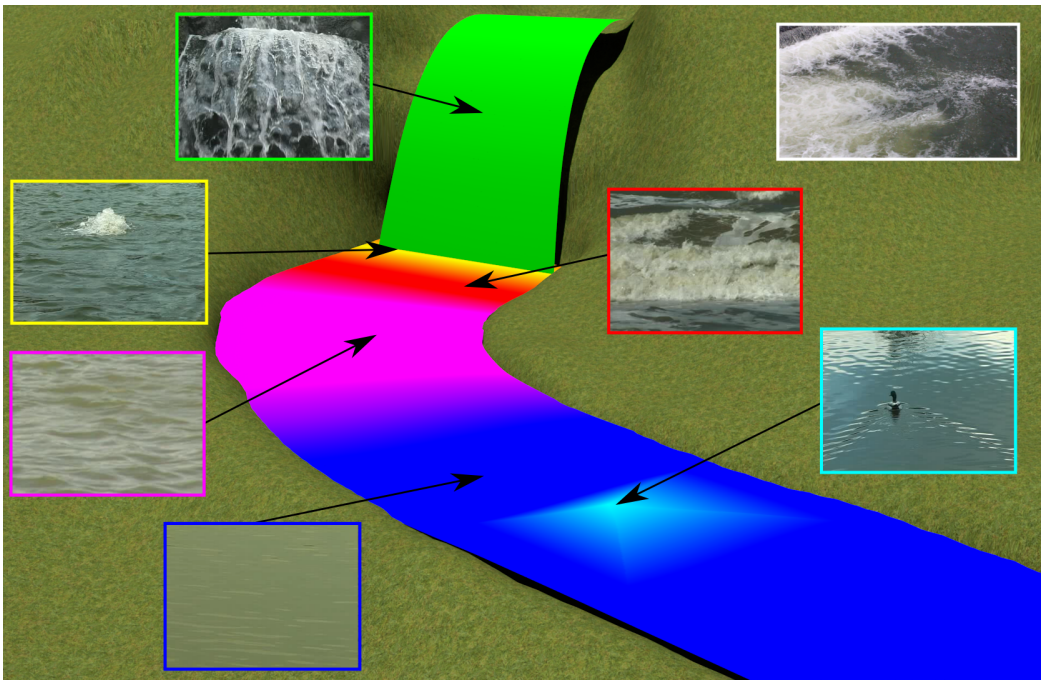
4.8 Conclusion

We have taken animated water surface patches output by our video reconstruction algorithm (Chapter 3) and developed methods of using them to create novel water animations for use within computer graphics applications. Our approach allows us to easily create water animations containing motion captured from multiple video examples, combined seamlessly together and mapped onto interesting meshes. Our method also includes the ability to add foam to the water surfaces. We have shown that we can produce high quality, visually plausible results. The output appearance depends only on the renderer, so can be manipulated by the user to suit the application. In contrast, the shape and movement are derived from the videos, which can be used to inspire the designer.

Although we use height fields for the water surface geometry, we have shown that we can visually approximate many non-height field phenomena, such as foam, waterfalls, splashes and boiling water. Our animations are unable to dynamically react to external forces, but we are able to apply forces to

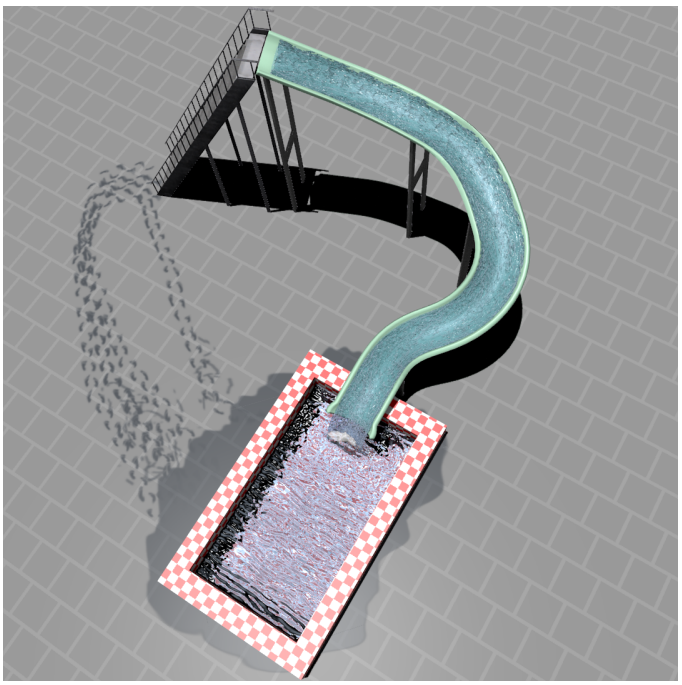
external objects by advection.

Rendered result

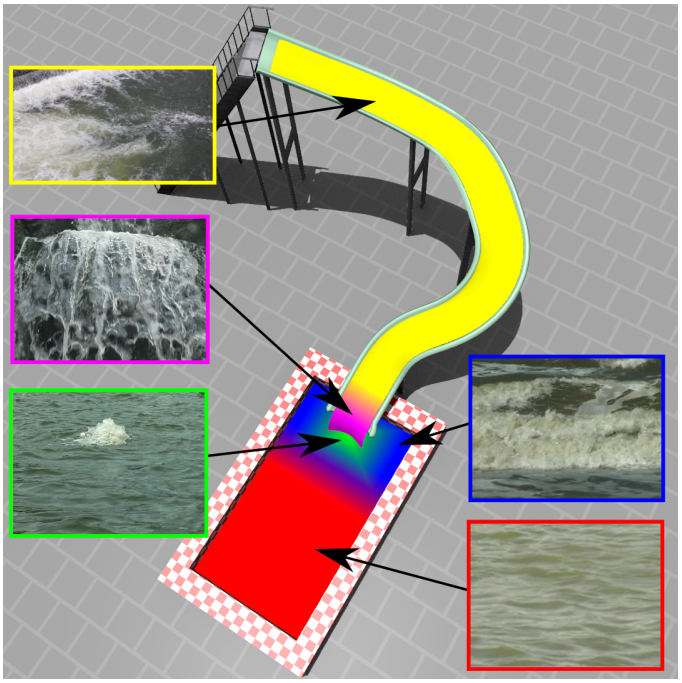


Mapping of examples

Figure 4.13: River scene. The video example highlighted in green was also used to create a foam texture for the same location. The example highlighted in white was used to create a foam texture for the lower river.



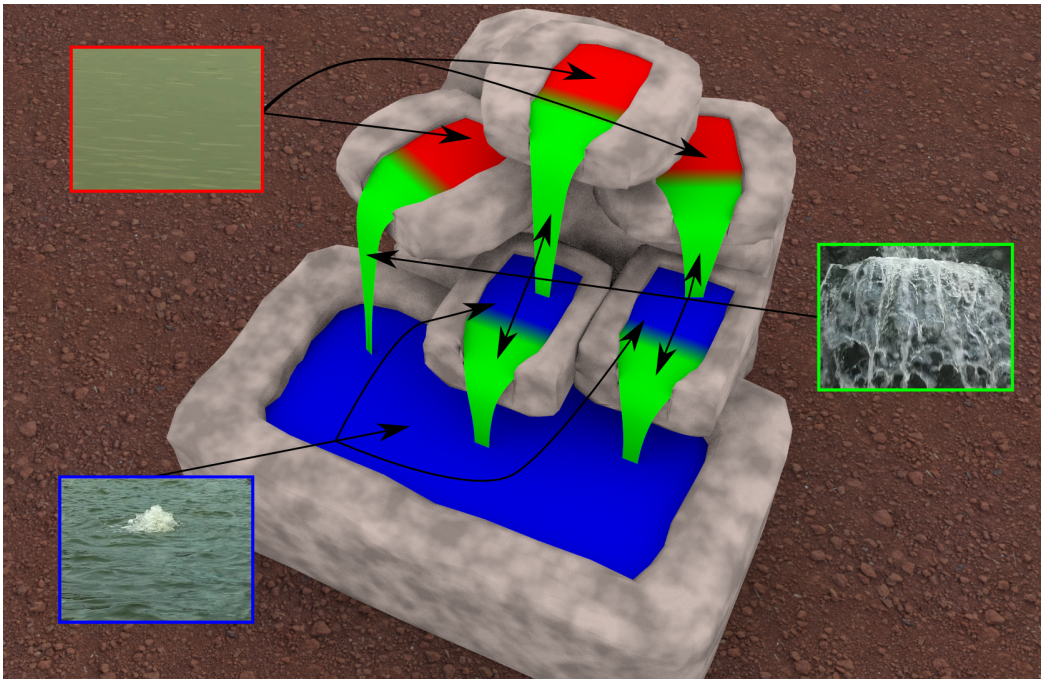
Rendered result



Mapping of examples

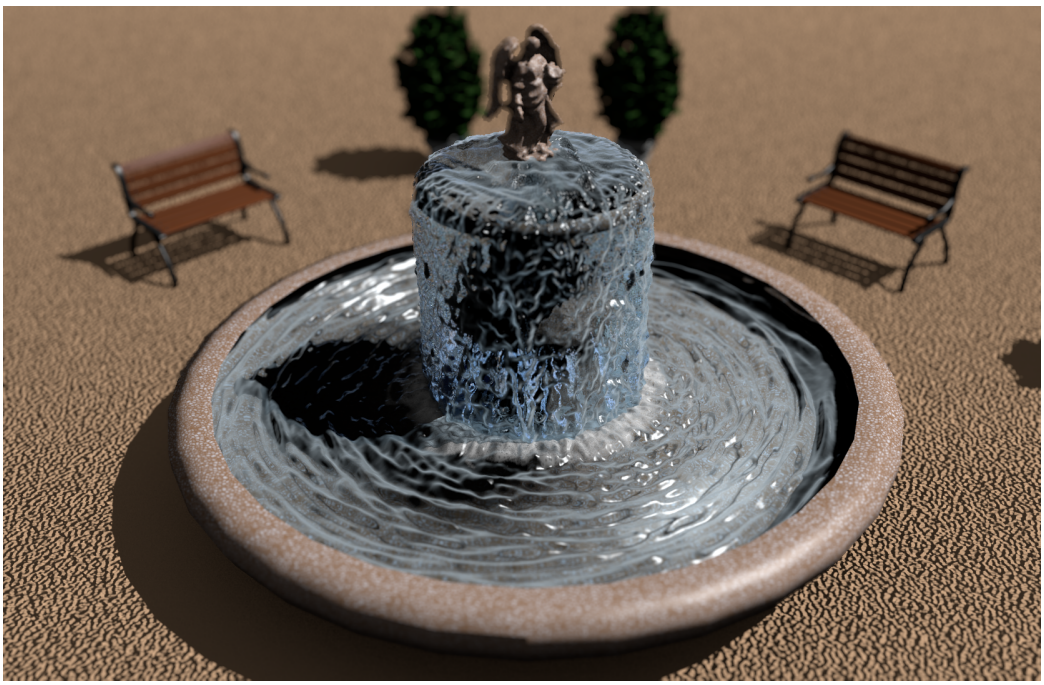
Figure 4.14: Water slide scene. The video example highlighted in green was also used to create a foam texture.

Rendered result

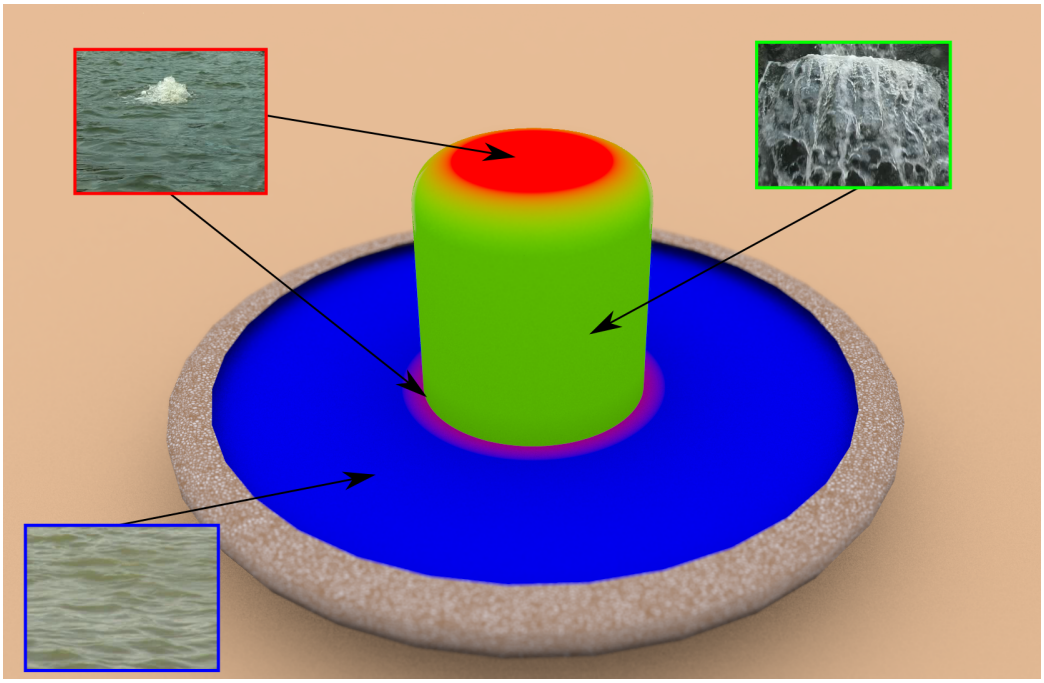


Mapping of examples

Figure 4.15: Water feature scene. The video example highlighted in blue was also used to create a foam texture.

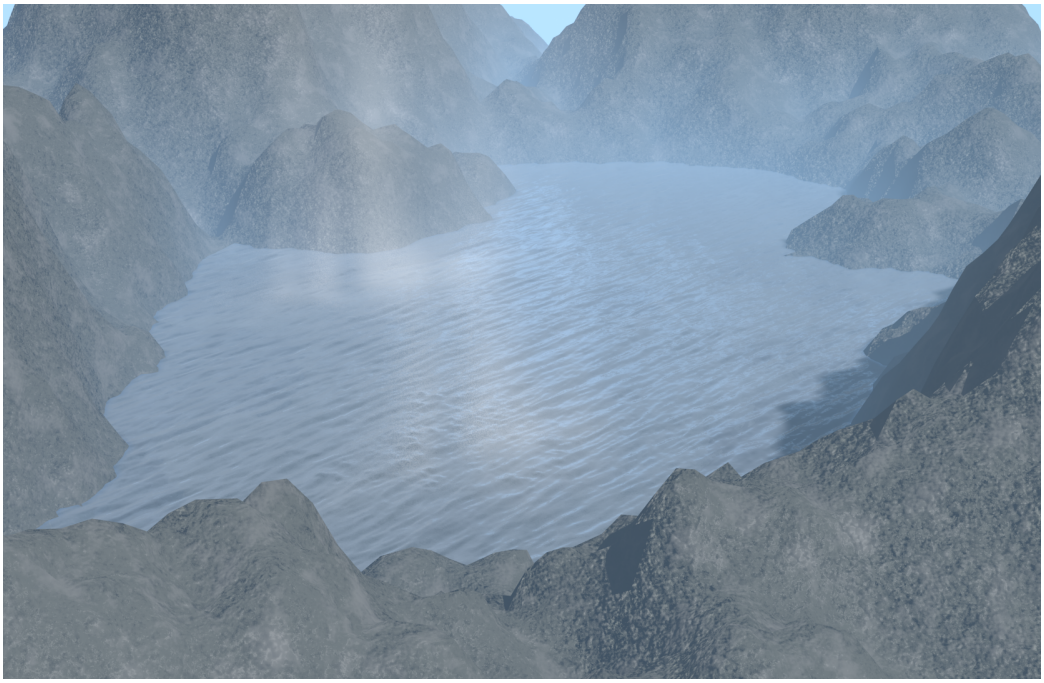


Rendered result

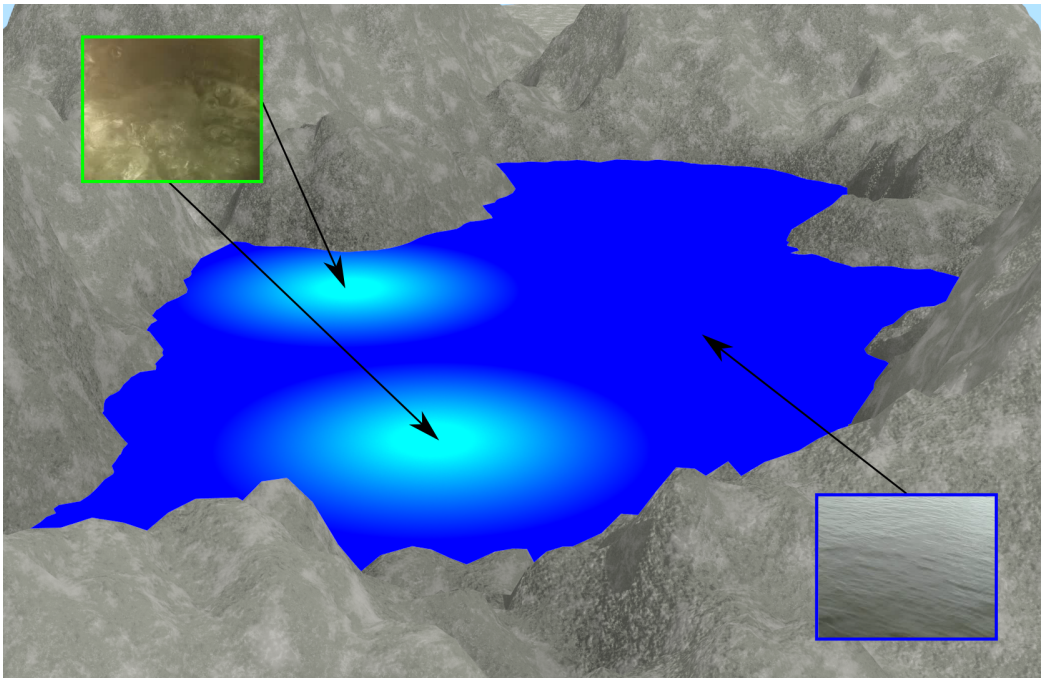


Mapping of examples

Figure 4.16: Fountain scene. The video example highlighted in red was also used to create a foam texture.

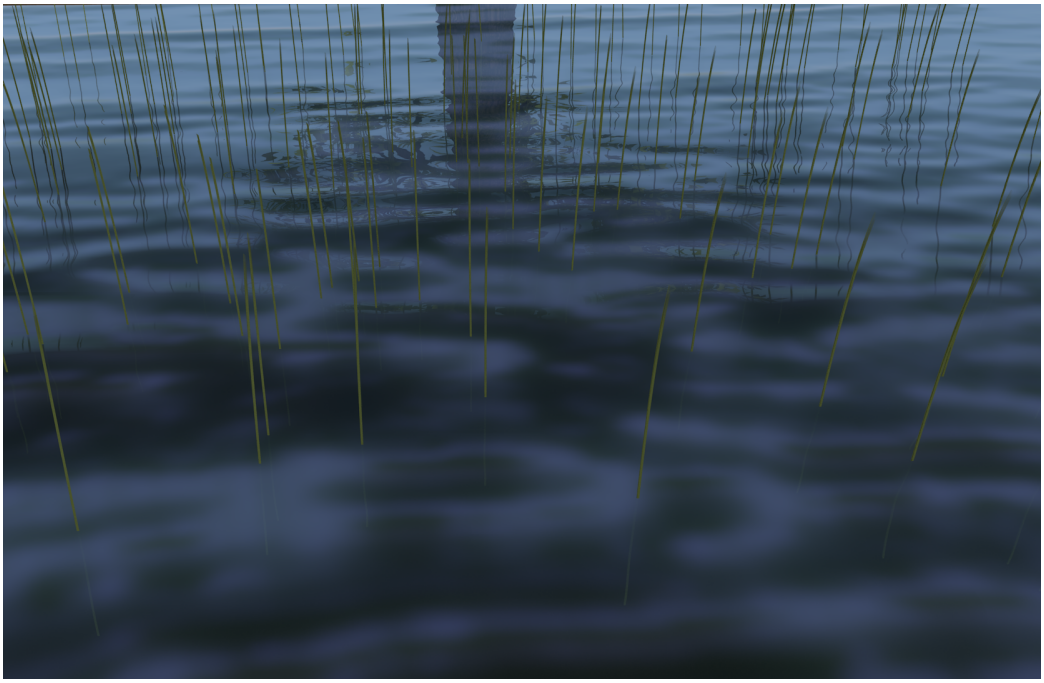


Rendered result



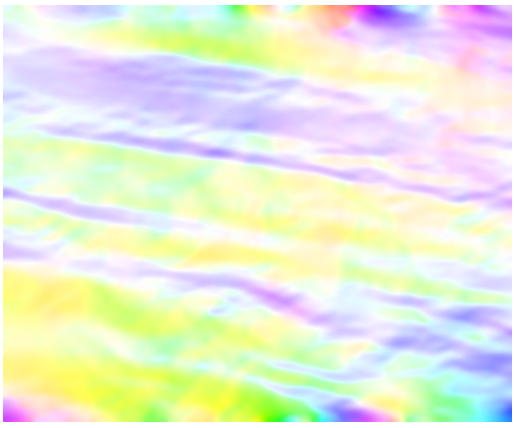
Mapping of examples

Figure 4.17: Hot springs scene.



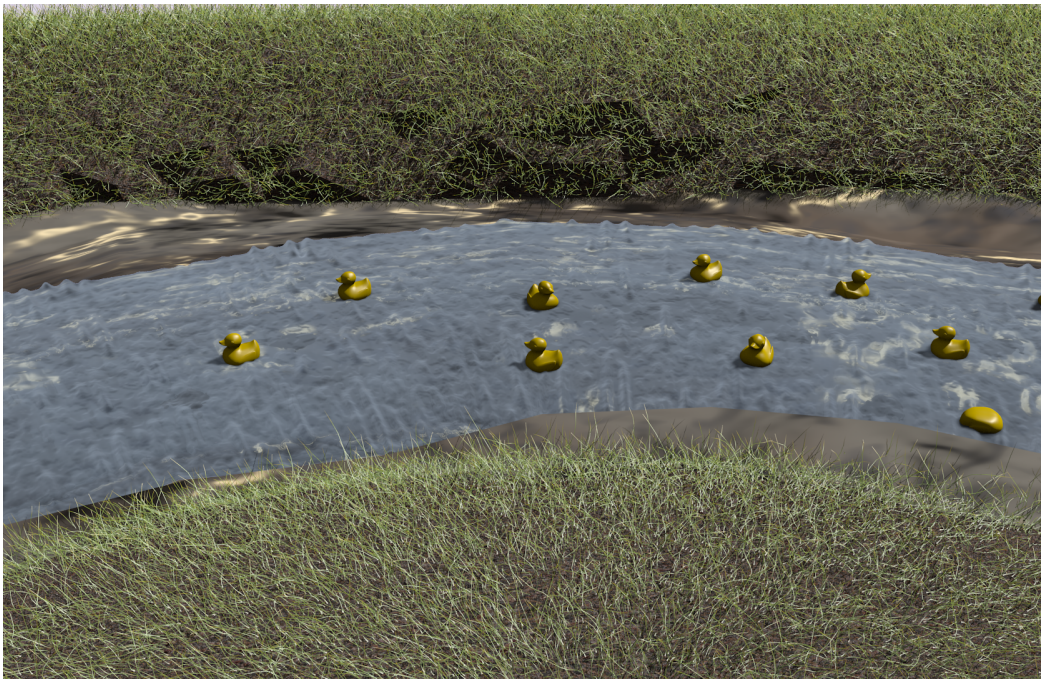
Rendered result

Video example



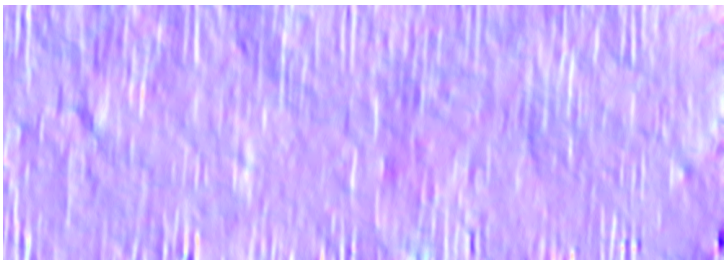
Velocity field

Figure 4.18: Lake with reeds scene. The velocity field was used to drive the movement of the reeds.



Rendered result

Video example



Velocity field

Figure 4.19: Stream with rubber ducks scene. The velocity field was used to drive the movement of the ducks.

Chapter 5

Conclusion

In this work we have successfully shown that videos of real water scenes can be used as input examples from which novel three-dimensional scenes can be created. We have shown how videos can be used as visual parameters, instead of the traditional use of physical parameters, to design the visible behaviour of novel water surface animations. Our results have demonstrated that we are able to plausibly emulate a wide range of real-world scenes, different from those from which the water characteristics were captured. The key advantage of our approach over existing techniques is that it is simple for a naive user to create novel water animations, without any knowledge of fluid dynamics.

Our approach is split into two main pieces of work. The first is a method of reconstructing the perceived water surface geometry from video footage of real scenes, in the form of time-varying height and velocity fields. Our approach uses a simple capture setup, which allows it to be practically applied to natural outdoor scenes. Our results show that we are successful at reconstructing a wide range of different water behaviours, including plausible approximations of non height field phenomena such as breaking waves, splashes, waterfalls and boiling water.

The second major section of our work is an approach able to use the water reconstructions we have created to produce novel animations. We have presented a method for blending over the seams between two water surfaces, allowing the reconstructed surfaces to be used as building blocks which can be tiled together within a novel scene. We have shown how foam textures can be extracted from the original source videos and applied to the resulting

animations. As a result of including velocity fields, we have been able to use these to drive the movement of external objects within the scene. We have presented plausibly realistic results for a wide range of scenes.

Our approach does not aim for physical accuracy and contains some obvious approximations, however, we are able to achieve visually plausible results. There may be certain elements of human perception which explain this. Ferwerda et al. [25] show that high frequency texture patterns applied to 3D surfaces can visually mask out errors in the underlying surface shape. The high frequency movement of the water in some of our animations may help to mask errors in the surface shape of individual frames. Koenderink et al. [45] show that although humans are good at perceiving the global overall shape of an object, they perform less well at perceiving the exact local change in depth across a surface. Todd and Reichel [91] studied how humans are able to perceive an underlying smooth surface from a projected pattern of surface contours. They show that these surface contours can exhibit certain types of noise without hindering the perception of the underlying surface. These two works may also provide an insight into why humans can tolerate inaccuracies in our results. We leave further study in this area as future work.

We also leave room for other interesting future work in this area. Firstly, although height fields are able to plausibly approximate a wide range of water behaviour, they do have their limits. Future work could explore the possible reconstruction of a more sophisticated surface representation, whilst trying to keep the capture method simple enough to be used on natural scenes. An alternative would be to keep the height field representation, but allow it to be coupled with a particle system in areas which contain breaking waves or splashes, similar to the work carried out in the simulation community [13, 88, 89].

Secondly, methods for allowing the user to adjust the behaviour of a water surface would be an interesting research direction, such as changing the choppiness or frequency of the waves in a visually intuitive manner. This would allow the user to reconstruct a surface which is close to what they want, and then refine it to more closely match their requirements. Approaches which allow more sophisticated editing of the velocity field, and how the velocities are affected when the surfaces are shaped within a scene, would also be an interesting area to explore.

We have shown that we are able to include the appearance of foam in our animations, by applying textures extracted from video. Although this enhances

the realism of our results, our method does not emulate the true appearance of natural foam. Our method uses a texture on the surface of the water, whereas real foam can exist throughout the fluid volume. Real foam also has different material properties to water, beyond just a different colour. Future work could investigate a more sophisticated way to represent and render the foam in the animations. Creating realistic foam is still an open problem in fluid animation research in general.

The water animations we produce are not able to dynamically react to external forces. This is something which can still only be accomplished when using simulations. A third direction of future research could be to automatically learn the physical parameters which would produce a simulation that matched an animation produced with our approach. Whilst completing the work presented in this thesis, we have also explored this idea. The difficulties are that we produce some animations which do not fit the shallow water model, and would therefore need to use a full Navier-Stokes based simulation. This is a very difficult problem, as there are a huge number of parameters to solve for. These include the liquid's viscosity, a starting velocity and density at every point within the liquid volume, and any external forces can be applied to any positions within the fluid volume. We have already discussed that existing methods of finding the parameters which attract a liquid to simple target key frames take days to compute for low-resolution simulation grids [56]. There is also the difficult challenge of finding the correct trade-off between matching the behaviour designed by the user and accurately interacting with scene elements.

A possible alternative approach to achieve interactions between the water surfaces and external objects could be to learn the appearance of such interactions from a set of example videos. Future work in this direction may be able to achieve results which are visually plausible, without having to directly simulate any physics.

Lastly, our opinion is that the most interesting direction of future work would be to explore how we can use what we have learnt with water to develop similar methods, or adapt our current method, to authoring animations of other kinds of fluids such as fire and smoke. Like water reconstruction, previous methods of reconstructing fire and smoke use sophisticated capture setups to attempt to reconstruct a physically accurate reconstruction (Chapter 2). Our aim for perceptual plausibility has resulted in a method which uses a far simpler capture setup for reconstructing water scenes, employing the same aims for reconstructing other fluids may result in the same success. A volu-

metric representation, such as a three-dimensional volume, may have to be employed to be able to represent other fluid types. Our method for blending across different height fields would easily extend to three-dimensional volumes, but only experiments would show how suitable this type of blending would be for different fluid types.

Appendix A

Shape from Shading

Here we briefly describe the workings of the shape from shading algorithm used in Section 3.2, but please see the paper by Tsai et al. for the full details [93]. This method is an iterative method, which calculates an updated height $h^n(x, y)$ for each pixel (x, y) from a previous estimate $h^{n-1}(x, y)$. The initial estimate $h^0(x, y)$ for all pixels is set to zero. The update equation is as follows

$$h^n(x, y) = h^{n-1}(x, y) + K^n(-f(h^{n-1}(x, y))). \tag{A.1}$$

The value of K^n is defined as

$$K^n = \frac{S^n_{x,y} M_{x,y}}{10^{-8} + S^n_{x,y} M^2_{x,y}}, \tag{A.2}$$

where

$$M_{x,y} = \frac{\mathrm{d}f(h^{n-1}(x, y))}{\mathrm{d}h(x, y)}, \tag{A.3}$$

$$S^n_{x,y} = E[(h^n(x, y) - h(x, y))^2)]. \tag{A.4}$$

Here E is the is the expectation operaton, which means that $S^n_{x,y}$ will approach zero as the program iterates towards the solution. This is more specifically calculated as

$$S^n_{x,y} = S^{n-1}_{x,y} - S^{n-1}_{x,y} K^n \frac{\mathrm{d}f(h^{n-1}(x, y))}{\mathrm{d}h(x, y)}, \tag{A.5}$$

where the value of $S^0_{x,y}$ is set to 0.01. The derivative $\frac{\mathrm{d}f(h^{n-1}(x,y))}{\mathrm{d}h(x,y)}$, used in Equations A.3 and A.5, is calculated as

$$\frac{\mathrm{d}f(h_{n-1}(x,y))}{\mathrm{d}h(x,y)} = -(\frac{p_s + q_s}{\sqrt{p^2+q^2+1}\sqrt{p_s^2+q_s^2+1}} - \frac{(p+q)(pp_s+qq_s+1)}{\sqrt{(p^2+q^2+1)^3}\sqrt{p_s^2+q_s^2+1}}), \tag{A.6}$$

where

$$p = \frac{\partial h}{\partial x} = h(x,y) - h(x-1,y), \tag{A.7}$$

$$q = \frac{\partial h}{\partial y} = h(x,y) - h(x,y-1), \tag{A.8}$$

$$p_s = \frac{\cos\tau \sin\sigma}{\cos\sigma}, \tag{A.9}$$

$$q_s = \frac{\sin\tau \sin\sigma}{\cos\sigma}. \tag{A.10}$$

The values τ and σ are the tilt and slant of the light source, which we set to zero for all our results.

Appendix B

Optical Flow Euler-Lagrange Equations

Here we define each of the terms of the Euler-Lagrange equations (Equations 3.19 and 3.20) used to calculate the water velocities in Chapter 3. The Euler-Lagrange equations in Equations 3.19 and 3.20 are defined as

$$\frac{\partial L}{\partial u} - \frac{\partial}{\partial x}\frac{\partial L}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial u_y} = 0, \tag{B.1}$$

$$\frac{\partial L}{\partial v} - \frac{\partial}{\partial x}\frac{\partial L}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial v_y} = 0, \tag{B.2}$$

where L is the integrand of the error functional and u_x, u_y, v_x ,and v_y are the derivatives of u and v with respect to x and y respectively.

The integrand L of the error functional from Chapter 3 is defined in full as

$$(E_x u + E_y v + E_t)^2 + \alpha^2(u_x^2 + u_y^2 + v_x^2 + v_y^2) + \beta^2(u_x + v_y + w_z)^2. \tag{B.3}$$

The individual terms of Equations B.1 and B.2 are defined as

$$\frac{\partial L}{\partial u} = 2E_x(E_x u + E_y v + E_t), \tag{B.4}$$

$$\frac{\partial}{\partial x}\frac{\partial L}{\partial u_x} = 2\alpha^2 u_{xx} + 2\beta(u_{xx} + v_{yx} + w_{zx}), \tag{B.5}$$

$$\frac{\partial}{\partial y}\frac{\partial L}{\partial u_y} = 2\alpha^2 u_{yy}, \tag{B.6}$$

$$\frac{\partial L}{\partial v} = 2E_y(E_x u + E_y v + E_t), \tag{B.7}$$

$$\frac{\partial}{\partial x}\frac{\partial L}{\partial v_x} = 2\alpha^2 v_{xx}, \tag{B.8}$$

$$\frac{\partial}{\partial y}\frac{\partial L}{\partial v_y} = 2\alpha^2 v_{yy} + 2\beta(u_{xy} + v_{yy} + w_{zy}). \tag{B.9}$$

We substitute these definitions into Equations B.1 and B.2, and remove the common multiplication factor of 2, to produce the following equations which are identical to Equations 3.21 and 3.22

$$E_x(E_x u + E_y v + E_t) - \alpha^2\nabla^2 u - \beta^2(u_{xx} + v_{yx} + w_{zx}) = 0, \tag{B.10}$$

$$E_y(E_x u + E_y v + E_t) - \alpha^2\nabla^2 v - \beta^2(u_{xy} + v_{yy} + w_{zy}) = 0. \tag{B.11}$$

Bibliography

[1] 3dmd. http://www.3dmd.com.

[2] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin, and Richard Szeliski. Panoramic video textures. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 821–827, New York, NY, USA, 2005. ACM.

[3] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 217–226, New York, NY, USA, 2001. ACM.

[4] Bradley Atcheson, Ivo Ihrke, Wolfgang Heidrich, Art Tevs, Derek Bradley, Marcus Magnor, and Hans-Peter Seidel. Time-resolved 3d capture of non-stationary gas flows. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 132:1–132:9, New York, NY, USA, 2008. ACM.

[5] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, MichaelJ. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92:1–31, 2011.

[6] Günther Balschbach, Jochen Klinke, and Bernd Jähne. Multichannel shape from shading techniques for moving specular surfaces. In Hans Burkhardt and Bernd Neumann, editors, *Computer Vision — ECCV'98*, volume 1407 of *Lecture Notes in Computer Science*, pages 170–184. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0054740.

[7] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *Visualization and Computer Graphics, IEEE Transactions on*, 7(2):120 –135, apr-jun 2001.

[8] Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. High-quality passive facial performance capture using anchor frames. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 75:1–75:10, New York, NY, USA, 2011. ACM.

[9] Kiran S. Bhat, Steven M. Seitz, Jessica K. Hodgins, and Pradeep K. Khosla. Flow-based video synthesis and editing. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 360–363, New York, NY, USA, 2004. ACM.

[10] R. Borgo, M. Chen, B. Daubney, E. Grundy, G. Heidemann, B. Höferlin, M. Höferlin, H. Leitte, D. Weiskopf, and X. Xie. State of the art report on video-based graphics and video visualization. *Computer Graphics Forum*, 31(8):2450–2477, 2012.

[11] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, 2008.

[12] Stephen Chenney. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 233–242, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[13] Nuttapong Chentanez and Matthias Müller. Real-time simulation of large bodies of water with small scale details. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 197–206, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[14] Nuttapong Chentanez and Matthias Müller. Real-time eulerian water simulation using a restricted tall cell grid. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 82:1–82:10, New York, NY, USA, 2011. ACM.

[15] Yung-Yu Chuang, Dan B Goldman, Ke Colin Zheng, Brian Curless, David H. Salesin, and Richard Szeliski. Animating pictures with stochastic motion textures. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 853–860, New York, NY, USA, 2005. ACM.

[16] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 287–294, New York, NY, USA, 2003. ACM.

[17] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 361–368, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[18] Yoshinori Dobashi, Yusuke Shinzo, and Tsuyoshi Yamamoto. Modeling of clouds from a single photograph. *Computer Graphics Forum*, 29(7):2083–2090, 2010.

[19] A. Doshi and A. G. Bors. Navier-stokes formulation for modelling turbulent optical flow. In *Proceedings of the British Machine Vision Conference*, pages 97.1–97.10. BMVA Press, 2007. doi:10.5244/C.21.97.

[20] Jean-Denis Durou, Maurizio Falcone, and Manuela Sagona. Numerical methods for shape-from-shading: A new survey with benchmarks. *Computer Vision and Image Understanding*, 109(1):22 – 43, 2008.

[21] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033 –1038 vol.2, 1999.

[22] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM.

[23] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers & Structures*, 83(6-7):479 – 490, 2005. ¡ce:title¿Frontier of Multi-Phase Flow Analysis and Fluid-Structure¡/ce:title¿ ¡xocs:full-name¿Frontier of Multi-Phase Flow Analysis and Fluid-Structure¡/xocs:full-name¿.

[24] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 736–744, New York, NY, USA, 2002. ACM.

[25] James A. Ferwerda, Peter Shirley, Sumanta N. Pattanaik, and Donald P. Greenberg. A model of visual masking for computer graphics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 143–152, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[26] N. Foster and D. Metaxas. Controlling fluid animation. In *Computer Graphics International, 1997. Proceedings*, pages 178 –188, jun 1997.

[27] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 23–30, New York, NY, USA, 2001. ACM.

[28] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471 – 483, 1996.

[29] Alain Fournier and William T. Reeves. A simple model of ocean waves. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 75–84, New York, NY, USA, 1986. ACM.

[30] Mashhuda Glencross, Gregory J. Ward, Francho Melendez, Caroline Jay, Jun Liu, and Roger Hubbold. A perceptually validated model for surface depth hallucination. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 59:1–59:8, New York, NY, USA, 2008. ACM.

[31] B. Goldlucke, I. Ihrke, C. Linz, and M. Magnor. Weighted minimal hypersurface reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(7):1194 –1208, july 2007.

[32] Diego Gutierrez, Francisco J. Seron, Jorge Lopez-Moreno, Maria P. Sanchez, Jorge Fandos, and Erik Reinhard. Depicting procedural caustics in single images. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 120:1–120:9, New York, NY, USA, 2008. ACM.

[33] Francis H. Harlow and J. Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[34] S.W. Hasinoff and K.N. Kutulakos. Photo-consistent 3d fire by flame-sheet decomposition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1184 –1191 vol.2, oct. 2003.

[35] S.W. Hasinoff and K.N. Kutulakos. Photo-consistent reconstruction of semitransparent scenes by density-sheet decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(5):870 –885, may 2007.

[36] Tim Hawkins, Per Einarsson, and Paul Debevec. Acquisition of time-varying participating media. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 812–815, New York, NY, USA, 2005. ACM.

[37] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 229–238, New York, NY, USA, 1995. ACM.

[38] V. Hilsenstein. Surface reconstruction of water waves using thermographic stereo imaging. In *Image and Vision Computing New Zealand*, pages 102 –107, 2005.

[39] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1–3):185 – 203, 1981.

[40] I. Ihrke, B. Goidluecke, and M. Magnor. Reconstructing the geometry of flowing water. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1055 –1060 Vol. 2, oct. 2005.

[41] Ivo Ihrke and Marcus Magnor. Image-based tomographic reconstruction of flames. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 365–373, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[42] Ivo Ihrke and Marcus Magnor. Adaptive grid optical tomography. *Graphical Models*, 68(5):484–495, September 2006.

[43] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '90, pages 49–57, New York, NY, USA, 1990. ACM.

[44] Erum Arif Khan, Erik Reinhard, Roland W. Fleming, and Heinrich H. Bülthoff. Image-based material editing. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 654–663, New York, NY, USA, 2006. ACM.

[45] JanJ. Koenderink, AndreaJ. Doorn, and AstridM.L. Kappers. Surface perception in pictures. *Perception & Psychophysics*, 52(5):487–496, 1992.

[46] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 277–286, New York, NY, USA, 2003. ACM.

[47] Ares Lagae and Philip Dutré. An alternative for wang tiles: colored edges versus colored corners. *ACM Trans. Graph.*, 25(4):1442–1459, October 2006.

[48] Anita T. Layton and Michiel van de Panne. A numerically efficient and stable algorithm for animating water waves. *The Visual Computer*, 18:41–53, 2002.

[49] C. Li, D. Pickup, T. Saunders, D. Cosker, D. Marshall, P. Hall, and P. Willis. Water surface modeling from a single viewpoint video. *Visualization and Computer Graphics, IEEE Transactions on*, PP(99):1, 2012.

[50] Chuan Li, Oliver Deussen, Yi-Zhe Song, Phil Willis, and Peter Hall. Modeling and generating moving trees from video. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 127:1–127:12, New York, NY, USA, 2011. ACM.

[51] Feng Li, Liwei Xu, P. Guyenne, and Jingyi Yu. Recovering fluid-type motions using navier-stokes potential flow. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2448 –2455, june 2010.

[52] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, July 2001.

[53] Zhouchen Lin, Lifeng Wang, Yunbo Wang, S.B. Kang, and Tian Fang. High resolution animated scenes from stills. *Visualization and Computer Graphics, IEEE Transactions on*, 13(3):562 –568, may-june 2007.

[54] G.A. Mastin, P.A. Watterberg, and J.F. Mareda. Fourier synthesis of ocean scenes. *Computer Graphics and Applications, IEEE*, 7(3):16 –23, march 1987.

[55] Wojciech Matusik, Matthias Zwicker, and Frédo Durand. Texture design using a simplicial complex of morphable textures. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 787–794, New York, NY, USA, 2005. ACM.

[56] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 449–456, New York, NY, USA, 2004. ACM.

[57] J J Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703, 2005.

[58] N.J.W. Morris and K.N. Kutulakos. Dynamic refraction stereo. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1573 –1580 Vol. 2, oct. 2005.

[59] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[60] Matthias Müller, Jos Stam, Doug James, and Nils Thürey. Real time physics: class notes. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pages 88:1–88:90, New York, NY, USA, 2008. ACM.

[61] H. Murase. Surface shape reconstruction of a nonrigid transparent object using refraction and motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:1045–1052, 1992.

[62] Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, Luc van Gool, and Werner Purgathofer. A Survey of Urban Reconstruction. In *EUROGRAPHICS 2012 State of the Art Reports*, pages 1–28. Eurographics Association, 2012.

[63] Yoshikazu Nakajima, Hiroshi Inomata, Hiroki Nogawa, Yohshinobu Sato, Shinichi Tamura, Kozo Okazaki, and Seiji Torii. Physics-based flow estimation of fluids. *Pattern Recognition*, 36(5):1203 – 1212, 2003.

[64] Makoto Okabe, Ken Anjyo, Takeo Igarashi, and Hans-Peter Seidel. Animating pictures of fluid using video examples. *Computer Graphics Forum*, 28(2):677–686, 2009.

[65] Makoto Okabe, Ken Anjyor, and Rikio Onai. Creating fluid animation from a single image using video database. *Computer Graphics Forum*, 30(7):1973–1982, 2011.

[66] Sylvain Paris, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. Hair photobooth: geometric and photometric acquisition of real hairstyles. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 30:1–30:9, New York, NY, USA, 2008. ACM.

[67] Darwyn R. Peachey. Modeling waves and surf. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 65–74, New York, NY, USA, 1986. ACM.

[68] David Pickup, Darren Cosker, Peter Hall, and Philip Willis. Statistically user controlled texture mixing. In *Conference on Visual Media Production*, CVMP 2011 short papers, 2011.

[69] Willard J. Pierson and Lionel Moskowitz. A proposed spectral form for fully developed wind seas based on the similarity theory of s. a. kitaigorodskii. *Journal of Geophysical Research*, 69(24):5181–5190, 1964.

[70] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40:49–70, 2000. 10.1023/A:1026553619983.

[71] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. Image-based plant modeling. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 599–604, New York, NY, USA, 2006. ACM.

[72] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 193–202, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[73] N. Ray, B. Lévy, H. Wang, G. Turk, and B. Vallet. Material space texturing. *Computer Graphics Forum*, 28(6):1659–1669, 2009.

[74] Roland Ruiters, Ruwen Schnabel, and Reinhard Klein. Patch-based texture interpolation. *Computer Graphics Forum*, 29(4):1421–1429, 2010.

[75] M.A. Ruzon and C. Tomasi. Edge, junction, and corner detection using color distributions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1281 –1295, nov 2001.

[76] H. Sakaino. Fluid motion estimation method based on physical properties of waves. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[77] H. Sakaino, Yutao Zhao, and Yuncai Liu. 3d photodynamic tool using a single 2d image with 3d velocity primitives and physical effects. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 986 –989, 28 2009-july 3 2009.

[78] Thomas Schlömer and Oliver Deussen. Semi-stochastic tilings for example-based texture synthesis. *Computer Graphics Forum*, 29(4):1431–1439, 2010.

[79] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 489–498, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[80] Lin Shi and Yizhou Yu. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 229–236, New York, NY, USA, 2005. ACM.

[81] Barbara Solenthaler and Markus Gross. Two-scale particle simulation. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 81:1–81:8, New York, NY, USA, 2011. ACM.

[82] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[83] Carsten Stoll, Juergen Gall, Edilson de Aguiar, Sebastian Thrun, and Christian Theobalt. Video-based reconstruction of animatable human characters. In *ACM SIGGRAPH Asia 2010 papers*, SIGGRAPH ASIA '10, pages 139:1–139:10, New York, NY, USA, 2010. ACM.

[84] Ping Tan, Tian Fang, Jianxiong Xiao, Peng Zhao, and Long Quan. Single image tree modeling. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 108:1–108:7, New York, NY, USA, 2008. ACM.

[85] J. Tessendorf. Simulating ocean surfaces. In *ACM SIGGRAPH 2001 course notes*, 2001.

[86] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 7–12, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[87] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. *Graphical Models*, 71(6):221 – 228, 2009. ¡ce:title¿2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006)¡/ce:title¿.

[88] N. Thürey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Gross. Real-time simulations of bubbles and foam within a shallow water framework. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 191–198, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[89] Nils Thurey, Matthias Muller-Fischer, Simon Schirm, and Markus Gross. Real-time breaking waves for shallow water simulations. In *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, pages 39 –46, 29 2007-nov. 2 2007.

[90] Nils Thürey, Ulrich Rüde, and Marc Stamminger. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 157–164, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[91] J. T. Todd and F. D. Reichel. Visual perception of smoothly curved surfaces from double-projected contour patterns. *Journal of experimental psychology: human perception and performance*, 16(3):665–674, 1990.

[92] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 716–723, New York, NY, USA, 2003. ACM.

[93] Ping-Sing Tsai and Mubarak Shah. Shape from shading using linear approximation. *Image and Vision Computing*, 12(8):487 – 498, 1994.

[94] Huamin Wang, Miao Liao, Qing Zhang, Ruigang Yang, and Greg Turk. Physically guided liquid surface modeling from videos. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 90:1–90:11, New York, NY, USA, 2009. ACM.

[95] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the Art in Example-based Texture Synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117, Munich, Allemagne, 2009. Eurographics Association.

[96] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[97] Martin Wicke, Matt Stanton, and Adrien Treuille. Modular bases for fluid dynamics. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 39:1–39:8, New York, NY, USA, 2009. ACM.

[98] Ruo Zhang, Ping-Sing Tsai, J.E. Cryer, and M. Shah. Shape-from-shading: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):690 –706, aug 1999.

[99] Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. Building volumetric appearance models of fabric using micro ct imaging. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 44:1–44:10, New York, NY, USA, 2011. ACM.